AFOSR-TR-97

0137

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	2/18/97	Final Technical Report: 6/93-5/96

4. TITLE AND SUBTITLE

Developing Guided Search 3.0
The Next Generation of a Model of Visual Search

5. FUNDING NUMBERS

F49620-~~97-1-0045~~
93-1-0407

2313/BS

6. AUTHOR(S)

Jeremy M. Wolfe, Ph.D.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Brigham and Women's Hospital
75 Francis Street
Boston, MA 02155

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFOSR-NL
110 Duncan Avenue, Suite B-115
Bolling AFB
Washington, DC 20332-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release;
distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes achievement in four areas: 1) Guided Search 3.0 (GS3) is the third generation of the Guided Search model of visual search behavior. It models new visual search data and proposes a relationship between covert deployment of attention and overt eye movements. 2) Research documents the effect of eccentricity in visual search, shows the interaction of those effects with other visual and non-visual loads, and incorporates eccentricity effects into GS3. 3) A retrospective analysis of 2500 search sessions provides an unprecedented statistical picture of human search behavior. 4) Research shows that published inferences based on individual differences in search performance must be evaluated with great caution. These differences prove to be statistically unreliable. Nevertheless, with proper methods, it is possible to find reliable individual differences in visual search.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Visual Attention, Visual Search	38
	16. PRICE CODE

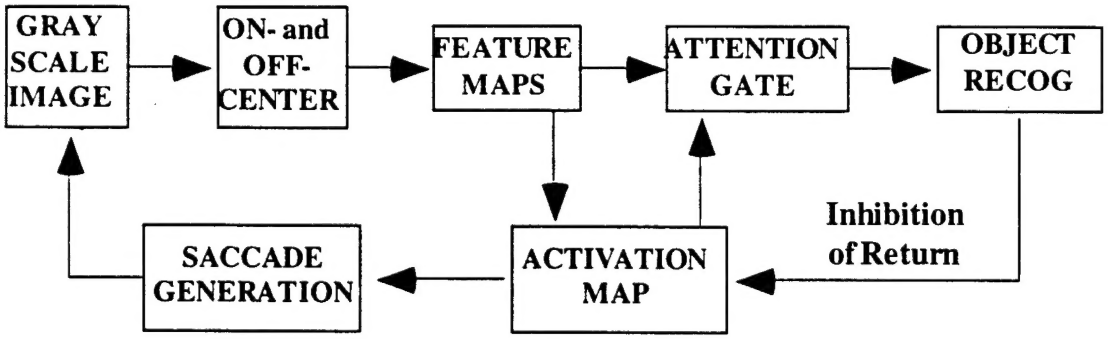
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
U	U	U	U

Developing Guided Search 3.0

AFOSR F49620-97-1-0045
Final Technical Report

Jeremy M Wolfe - PI

February, 1997



CONTENTS

Executive Summary

Four areas of achievement are described:

1) The Development of the Guided Search 3.0 (GS3) Model of Visual Search: GS3 is the next generation of Guided Search. It models new visual search data and proposes a relationship between covert deployment of attention and overt eye movements.

2) Eccentricity Effects in Visual Search: This project documents the effect of eccentricity in visual search, shows the interaction of those effects with other visual and non-visual loads, and incorporates eccentricity effects into GS3.

3) What 1,000,000 Trials Tell Us About Visual Search: A retrospective analysis of 2500 search sessions provides an unprecedented statistical picture of human search behavior.

4) Individual Differences in Visual Search: This project shows that published inferences based on individual differences in search performance must be evaluated with great caution. These differences prove to be statistically unreliable. Nevertheless, with proper methods, it is possible to find reliable individual differences in visual search.

Personnel involved in the research effort

Jeremy M Wolfe - PI
Greg Gancarz - Graduate Student (funded through subcontract to Boston U.)
Patricia O'Neill - Post-doctoral Fellow
Todd Horowitz - Post-doctoral Fellow
Sara Bennett - Research Assistant
Nikki Klempen - Research Assistant
Kari Dahlen - Research Assistant

Publications supported in whole or in part by AFOSR F49620-97-1-0045

Wolfe, J. M. (1993). Guided Search 2.0: The Upgrade. Proc. of the Human Factors and Egonomics Society, 37th annual meeting, (Seattle, WA) pp 1295-1299.

Wolfe, J. M. (1994). Visual search in continuous, naturalistic stimuli. *Vision Research*, 34(9), 1187-1195.

Wolfe, J. M. (1994). The pertinence of research on visual search to radiologic practice. *Academic Radiology*, 2, 74-78

Bilsky, A. A., & Wolfe, J. M. (1995). Part-whole information is useful in size X size but not in orientation X orientation conjunction searches. *Perception and Psychophysics,* 57 (6), 749-760

Wolfe, J. M. (1996). Extending Guided Search: Why Guided Search needs a preattentive "item map". In A. Kramer, G. H. Cole, & G. D. Logan (Eds.), Converging operations in the study of visual selective attention., (pp. 247-270). Washington, DC: American Psychological Associatio

Wolfe, J. M., & Gancarz, G. (1996). Guided Search 3.0: A model of visual search catches up with Jay Enoch 40 years later. In V. Lakshminarayanan (Ed.), Basic and Clinical Applications of Vision Science, . Dordrecht, Netherlands: Kluwer Academic..

Wolfe, J. M., & Bennett, S. C. (1997). Preattentive Object Files: Shapeless bundles of basic features. *Vision Research* 37, 1, 25-44

Wolfe, J. M., O'Neill, P. E., & Bennett, S. C. (1997). Why are there eccentricity effects in visual search? *Perception and Psychophysics,* in press

Wolfe, J.M. (1997) Visual Search in Pashler, H (ed) Attention, MIT Press, in press

O'Neill, P., & Wolfe, J. M. (1997). Individual Differences in Visual Search or Why testing two authors and one naive subject is inadequate. submitted

Wolfe, J M (1997) Post-attentive vision. *submitted*

Wolfe, J M (1997) What do 1,000,000 trials tell us about visual search? *submitted*

Overview

This report will describe four areas of acheivement in some detail:

1) The Development of the Guided Search 3.0 (GS3) Model of Visual Search: A central aim of this project has been the revision and expansion of the Guided Search model (Wolfe, Cave & Franzel, 1989, Wolfe, 1994). GS3 models new visual search data and proposes a relationship between covert deployment of attention and overt eye movements. A brief published description of GS3 is found in Wolfe and Gancarz (1996). The c-code and documentation for the simulation are attached as Appendix 1.

2) Eccentricity Effects in Visual Search: Earlier versions of the Guided Search simulation had pretended that the retina was homogeneous. GS3 has a fovea and a cortical magnification factor built into it. Since the published data on search and eccentricity were inadequate for our purposes, we have performed a series of experiments looking at the deployment of attention as a function of eccentricity. They show that, even in serial search, deployment of attention is systematic. One paper based on these experiments is in press (Wolfe, O'Neill & Bennett, 1997). A second paper awaits the results of on-going experiments.

3) What 1,000,000 Trials Tell Us About Visual Search: The goal of GS3 is to model human visual search behavior. As we worked on the development of the GS3 model, it became clear that we did not have an adequate picture of the behavior we wanted to model. Accordingly, we went back to a decade of visual search experiments and extracted 2500 sessions, each of one subject running several hundred trials of some visual search task. From the RT x set size slopes for target-present and target-absent trials, we can provide an unprecedented statistical picture of human search behavior. The new information from this analysis will constrain the GS3 model. A manuscript decribing this work has been submitted for publication (Wolfe, 1997c).

4) Individual Differences in Visual Search: A second goal of this project has been the evaluation of individual differences in visual search. Thus far, our most important finding in this area has been negative. Individual differences, based on running each subject on the usual 300-400 trials, are unreliable. A manuscript decribing this work has been submitted for publication (O'Neill & Wolfe, 1997).

Introduction to Visual Search

What is visual attention for?

Visual attention is part of the solution to a difficult problem in visual perception. The visual system is capable of great feats of analysis but it cannot do everything, everywhere, all the time. For instance, humans are very good at face recognition, but the evidence suggests that they can recognize only one face at a time (Nothdurft, 1993b; Suzuki & Cavanagh, 1995). Face recognition involves understanding the relationship between eyes, nose, mouth and other components (e.g. Reinitz, Morrissey & Demb, 1994) and the module responsible for face recognition seems to want two eyes, one nose, and one mouth as input. Without selective attention, this module might incorrectly attempt to *bind* together pieces from different faces when multiple faces are present in the scene. Selective attention can gate the input so that only one face at a time reaches the face recognition module. Face recognition is merely an example of the general problem of selection in visual processing. Similar considerations limit reading to just one stream of text at one time, even if multiple streams are present in the visual field. Even recognition of the shape of objects seems to proceed one object at a time (Wolfe & Bennett, 1997). In each case, visual attention is used to limit some aspects of visual processing to only a portion of the visual stimulus.

Other aspects of visual processing are not limited. They can be considered to be preattentive. 'Preattentive' is generally used to refer to processes thought to lie earlier in visual processing than the point at which attention intervenes. This may be misleading since this sequence is not logically required. Processes that do not require an attentional gatekeeper to restrict their input could, for instance, lie in parallel with attentive processes. Nevertheless, we will use the conventional term "preattentive".

Why not process everything preattentively, in parallel? In part, the answer lies in constraints imposed by neurophysiology. With an array of photoreceptors, it is possible to catch photons preattentively, in parallel, across the entire visual field. Similarly, one can perform a limited number of other operations in parallel (see the discussion of preattentive "features" below). However, with a brain of finite size, it is physically impossible to process all of the relationships between simple features in parallel (e.g. Is the vertical thing red?), let alone perform the complex calculations needed to identify your grandmother (Tsotsos, 1990). Consequently, the visual system has adapted a strategy in which analysis of simple visual properties like color or orientation is carried out in parallel by a large set of cells with relatively small, spatially localized receptive fields while more complex tasks like face recognition are performed by fewer cells with very large receptive fields (Gross, Bender & Gerstein, 1979; Schwartz, Desimone, Albright & Gross, 1983). While such a cell with a large receptive field might respond to a monkey hand anywhere in one hemifield, the cell might become confused if multiple hands or multiple objects were present in the field. The role of attention, described at a physiological level, may be to dynamically restrict the receptive field to a single item of interest (e.g. Chelazzi, Miller, Duncan & Desimone, 1993; Cowey, 1993; Desimone & Duncan, 1995; Moran & Desimone, 1985b).

Two stages of visual processing

Thirty years ago, Neisser (1967) introduced the idea of the two-stage visual system architecture that has provided the foundation for most subsequent models of visual attention. In the first stage, the preattentive stage, a limited number of basic visual features are processed in parallel across the entire visual field. This set of basic features includes color, orientation, size, and motion, in addition to some more complex attributes such as a variety of cues to depth and form. The evidence for a set of a dozen or so basic, preattentive features is reviewed in Wolfe (1997b). The second, attentive stage of processing is capable of more complex perceptual operations, but it is limited to performing those operations on one or, perhaps, a few items at a time. These two stages are sometimes referred to as the "parallel" stage and the "serial" stage, but this is somewhat of a misnomer, particularly for the attentive stage. "Serial" processes like face recognition and reading undoubtedly have some parallel characteristics. Eyes, nose, and mouth probably enter into face recognition in parallel. The stage is serial in the sense that the processes in this stage can be applied to only one item or one location at a time.

Attention can restrict processing to either specific locations or specific objects. Earlier models had assumed that attention selected a region of space but recent work shows that attention can be directed to objects (Baylis, 1994; Baylis & Driver, 1993; Behrmann & Tipper, 1994; Tipper, Weaver, Jerreat & Burak, 1994; Vecera & Farah, 1994).[1] In order to have object-based attention, some preattentive representation of objects must exist (Wolfe & Bennett, 1997). In the last few years, work in our lab and elsewhere has begun to examine the properties of these preattentive objects (Bilsky & Wolfe, 1995; Rensink, O'Regan & Clark, 1995; Wolfe & Bennett, 1996; Wolfe, Friedman-Hill & Bilsky, 1994).

[1]It is not necessary or correct to conceive of attention as strictly object-based or strictly space-based. It is possible, in the same display to instruct subjects to direct their attention to either a location or an object (e.g. Gibson & Egeth, 1994).

Our recent work indicates that the preattentive representation of an object is analogous to a child's model plane when it is still in the box. All the parts are there but the shape of the whole and the relationships between its parts cannot be appreciated until some attention is devoted to putting the pieces together (Wolfe & Bennett, 1997). We are now investigating what happens to that 'model' once attention moves elsewhere. We call this the problem of post-attentive vision. Interestingly, we now have data suggesting that the post-attentive visual representation is essentially the same as the preattentive representation. The model does not stay glued together but, when attention goes elsewhere, the pieces go back into the box (Wolfe, 1997a).

The Visual Search Paradigm

Our primary tool for studying visual attention has been visual search - the ability to find a target item in a visual field containing other, distracting items. Visual search tasks are ubiquitous in the real world. Drivers search for road signs (Ball, Owsley, Sloane, Roenker & Bruni, 1993). Radiologists search for tumors (Nodine, Krupinski & Kundel, 1993; Swensson & Judy, 1981; Wolfe, 1994b). Pilots search for landmarks. In the laboratory, simplified search tasks are employed. Subjects are presented with an array of items on a computer screen and asked to press one key if a target item is present and another key if no target is present. The usual dependent measures of interest are the reaction time (RT) and the accuracy of the response.

Building on Neisser's two-stage conception, it was once thought that performance on search tasks could be characterized as either "parallel" or "serial" on the basis of the RT results. For example, consider a *feature search* for a vertical item among horizontal distractors as shown in Figure 1a:

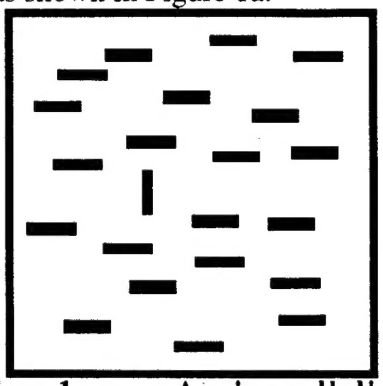


Fig 1a - A 'parallel' feature search for vertical among horizontals.

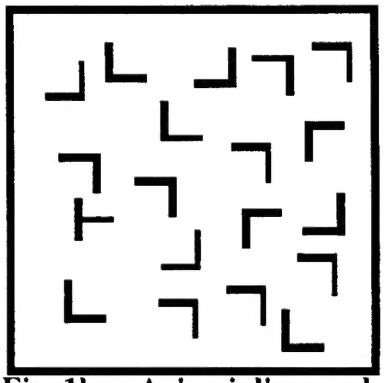


Fig 1b - A 'serial' search for a T among L's

If we vary the number of distracting items (varying "set size"), RT does not change (See Fig 3). The slope of the RT x set size function is near 0.0 msec/item for both target present and target absent trials suggesting that all items can be processed in a single step. By contrast, Figure 1b shows a sample of a search for a T among Ls of various orientations. This search appears to be serial and self-terminating (Kwak, Dagenbach & Egeth, 1991a). The slopes of the RT x set size functions are 20-30 msec/item for target present trials and 40-60 msec/item for blank trials, suggesting that serial process that can handle approximately 20 items per second. The claim that attention can be deployed at a rate of about one item every 50 msec is controversial (Duncan, Ward & Shapiro, 1994; Ward, Duncan & Shapiro, 1996), but we have recently found new evidence in support of this brief attentional "dwell time" (Bennett & Wolfe, 1996) as has Rensink (personal communication).

The problem with the dichotomous, serial-parallel account of visual search performance

is that search data actually fall on a continuum (Wolfe, 1997c). It is true that simple *feature searches* produce shallow slopes as long as targets and distractors are sufficiently different (e.g. Bauer, Jolicœur & Cowan, 1996; Duncan & Humphreys, 1989; Nagy & Sanchez, 1990). It is also the case that slopes are consistent with serial search when targets and distractors share the same basic features as in Fig. 1b. However, many tasks produce intermediate results. The most important class of such tasks are *conjunction searches* - searches where the target is defined by some combination of two or more basic features. Figure 2a shows an example. A search for a black vertical item among black horizontal and white vertical distractors cannot be done on the basis of color or orientation information alone. Treisman had once proposed that all such searches were serial (Treisman & Gelade, 1980), but subsequent research has shown that when salient stimuli are used, conjunction searches can be quite efficient, often with target present slopes in the range of 5-10 msec/item[2] (Dehaene, 1989; Egeth, Virzi & Garbart, 1984; Nakayama & Silverman, 1986; Quinlan & Humphreys, 1987; Theeuwes, 1995; Treisman & Sato, 1990; Wolfe, Cave & Franzel, 1989).

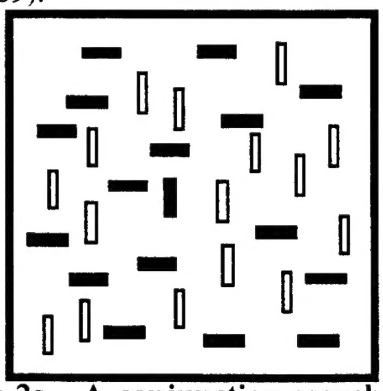


Fig 2a - A conjunction search for black vertical.

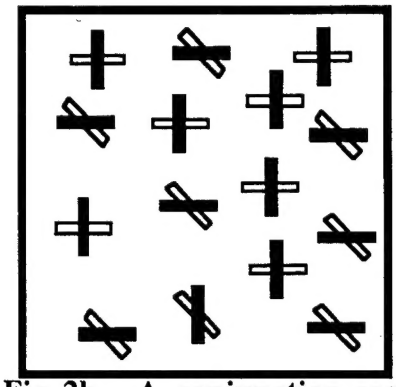


Fig 2b - A conjunction search for vertical and oblique.

At the far end of the continuum, it is worth noting that there are search tasks that produce slopes steeper than those seen in a standard "serial" search. An example is shown in Fig 2b. We have found that conjunctions of two of the same class of features - two colors, two orientations - yield very inefficient search with slopes of, perhaps, twice the usual "serial" slopes (Wolfe et al., 1990). Thus, it is probably advisable to drop the effort to divide search tasks into categories of "serial" and "parallel" and, instead, describe specific results on a continuum from "efficient" to "very inefficient" based on the slopes of RT x set size functions. This is illustrated in Figure Three.

[2] Note that virtually all natural search tasks are conjunction searches. One rarely searches for "green". One searches for the green book. At the other extreme, a search for the May '96 issue of a journal is, no doubt, serial through the set of journals but it is not serial through the entire scene. One would not search the view out the window for that journal.

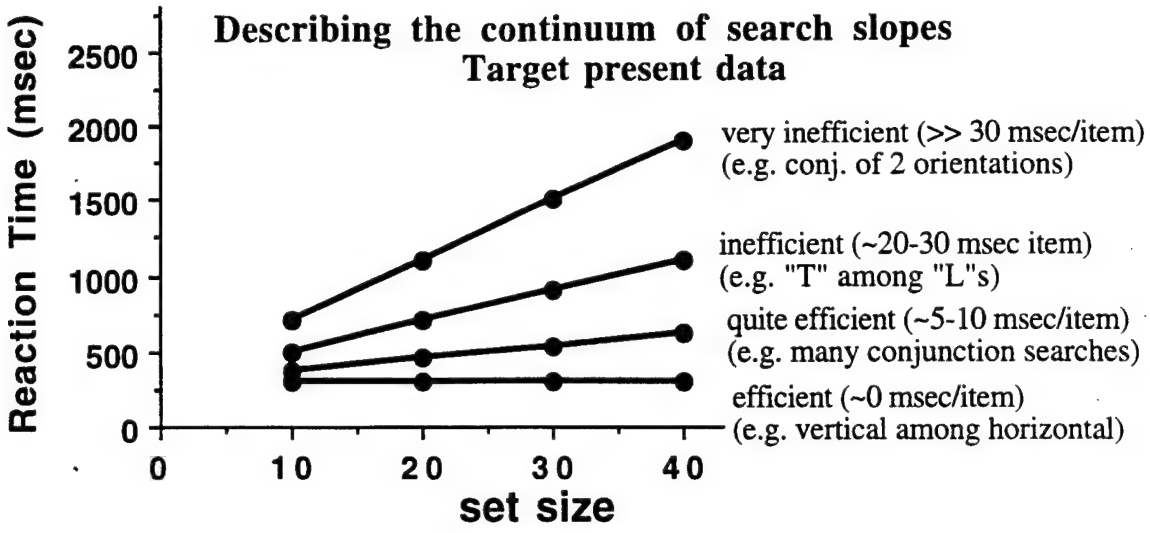


Figure Three: Results of visual search tasks fall on a continuum from efficient to very inefficient search (Functions are idealized from published results from our lab.)

To summarize, visual attention is an indispensable aspect of normal visual perception. By restricting processing to a single location or item, attention allows limited-capacity perceptual mechanisms to function. The role of attention can be studied with visual search tasks. It had been proposed that search tasks could be divided into two groups. Tasks in the preattentive group would be performed without attention, while others would require the random deployment of attention to item after item in a serial, self-terminating manner. Subsequent data did not support this strict dichotomy. We have worked for about ten years to develop a more adequate account of the deployment of visual attention. The result is our Guided Search model. The portion of this introduction describes the basic ideas of Guided Search as set forth in the original Guided Search paper (Wolfe et al., 1989) and its revision, Guided Search 2.0 (Wolfe, 1993). After that, we turn to the description of the specific projects funded by this grant.

Guided Search

The central idea of Guided Search (henceforth GS) is that the parallel, preattentive processes can be used to guide the deployment of attention. That is, even if a preattentive process does not represent the information needed to find a target in a visual search task, it may have partial information that can be used to deploy attention in a meaningful, rather than a random, manner from item to item. Consider the conjunction search example shown in Figure Four. No preattentive processor alone can mark the location of the black vertical item. However, an orientation processor can mark all the vertical items and a color process can mark all the black items. When these two sources are brought together in an "activation map", a black vertical item will be marked by the intersection of the two preattentive sources. If the deployment of attention is guided by this activation map, then attention will be efficiently deployed to the target.

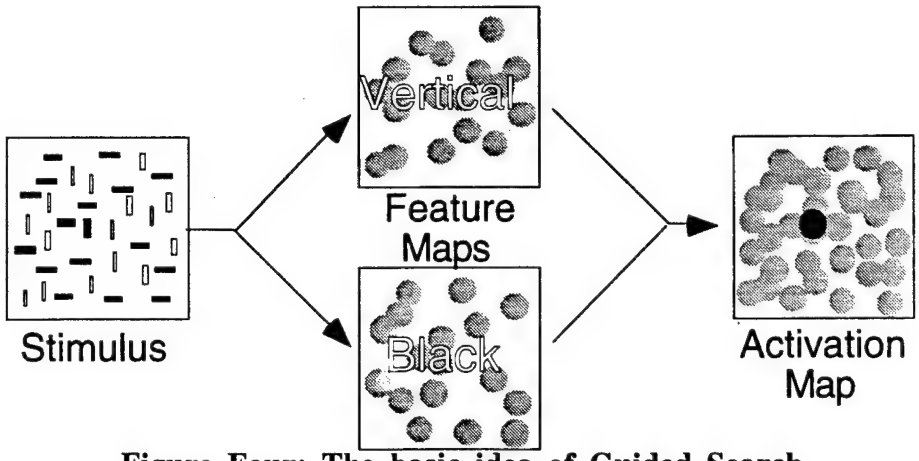


Figure Four: The basic idea of Guided Search

The roots of the idea of guidance can be found in some earlier work from other laboratories (Egeth et al., 1984; Hoffman, 1979) and guidance has now been incorporated into models other than Guided Search (e.g. Cohen & Ivry, 1991; Driver, 1992; Treisman & Sato, 1990).

Types of Guidance

Top-down Guidance

We can distinguish between two broad classes of attentional guidance: top-down and bottom-up. The guidance described in the preceeding example is top-down user-driven guidance. In this case, top-down control allows "black" and "vertical" information to be summed into the activation map while other, irrelevant information (e.g. red, curved) would be excluded. We have worked to understand the limits on top-down control of attentional guidance and have uncovered several basic principles:

1. The activation map can receive input from multiple features. Two features are required for simple conjunctions. Three features define triple conjunction search, which can be *more* efficient than standard conjunctions (Wolfe et al., 1989).

2. It is not possible to activate items based on the *absence* of a specific feature value. That is, one cannot command the color processor to activate all items that are not red (Friedman-Hill & Wolfe, 1995).

3. Activation of a feature like "red" could be implemented in the nervous system by excitation of red items or inhibition of 'not red' items. The most recent evidence indicates that activation in conjunction search involves both excitation of items with target attributes and inhibition of items without target attributes (Horowitz & Wolfe, 1996; see also Kim & Cave, 1995).

4. Conjunctions within a dimension like color or orientation are generally very inefficient. Thus, while it is possible to guide attention to an item that is red and vertical or big and green, it is not possible to guide attention to the item that is red and green or vertical and oblique (Fig. 2b) (Wolfe et al., 1990).

5. While it is not possible to guide attention to the item that is red and green, it is possible to guide attention to the whole red item with a green part - suggesting that preattentive representations contain some information about the part-whole structure of items (Bilsky & Wolfe, 1995; Wolfe et al., 1994).

6. Top-down specification of stimulus attributes appears to be *categorical*. For instance, for purposes of the guidance of attention, orientations are either "steep" or "shallow" and either tilted "left" or "right". Efficient search is possible for a stimulus that is categorically unique (e.g. the only "steep" item) but not for a stimulus that is not categorically unique (e.g. the item that is steep and tilted right among distractors that are either steep and left or shallow and right.) (Wolfe, Friedman-Hill, Stewart & O'Connell, 1992).

Bottom-up Guidance

In addition to top-down, user-driven guidance, there is bottom-up, stimulus driven guidance. Locations of abrupt change in the stimulus tend to attract attention as seen in Figure Five.

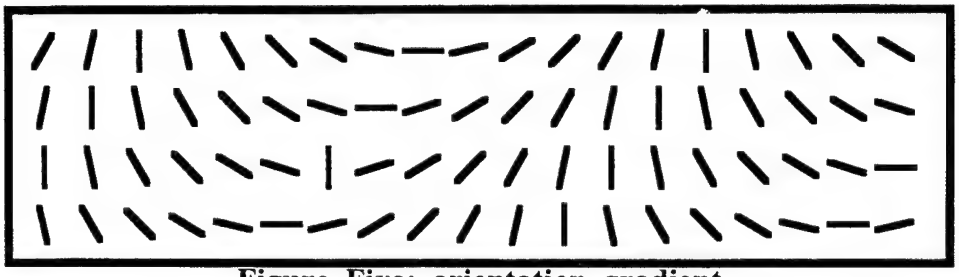


Figure Five: orientation gradient

One vertical line "pops out" even though you are not instructed to look for vertical and even though there are seven other vertical lines in the display (Moraglia, 1989; Nothdurft, 1985). The strength of "pop-out" depends on the local differences between a stimulus and its neighbors. It also depends on the physical distance between items. Julesz speaks of this in terms of a "texton gradient" (Julesz, 1986; Julesz & Bergen, 1983). Considerable research has been devoted to studying the mandatory capture of attention by pop out stimuli (reviewed in Wolfe, 1996b and Yantis, 1993). Introduction of a new object seems to be the strongest candidate for an event that will inevitably attract attention (Yantis & Gibson, 1994). More generally, the more dramatic the local change, the more likely attention will be attracted to that location/object. There is some evidence that pop-out is modulated by a top-down factor of its own. That is, singular events are more likely to capture attention if you know that you are looking for singular events (Bacon & Egeth, 1994, see also Mack, Tang, Tuma & Kahn, 1992). Indeed, top-down and bottom-up guidance can interact in quite complex ways. Consider a display with red items of orientation X and green items of orientation Y (where X and Y vary from trial to trial). A target that is red and of orientation Y will not pop out of a display with many red items and many items of orientation Y. It can be made to pop-out by the top-down instruction that the target is the odd item in the red subset. In this case, top-down guidance makes bottom-up guidance possible (Friedman-Hill & Wolfe, 1995).

The Guided Search 2.0 Simulation

Many of these principles were simulated in our GS2 simulation (Wolfe, 1994a). It could successfully perform searches for stimuli defined by color, orientation, or both. Bottom-up activation was calculated based on local differencing operations for each feature. Top-down activation was categorical and was separate for each feature. An overall, attention-guiding activation map was created by taking a weighted average of the activity in the feature maps. The weighting allowed for task demands to influence the creation of the overall activation map. Thus, in a color search, orientation information could be largely ignored and in a conjunction search, irrelevant bottom-up information could be largely ignored. The activation map was degraded by internal noise.

Attention was deployed from location to location on the basis of values in the activation map. Attention first went to the locus of greatest activation and then to other items in descending order. Search was terminated when the target was found or when no more items remained with activation levels above a dynamic threshold (Chun & Wolfe, 1996). Search efficiency is, in effect, a signal detection problem. The target gets some amount of attention-guiding activation (signal). Distractor items also have some activation (noise). In efficient "pop-out" searches, the signal is so much larger than the noise that attention is always deployed to the target first. In inefficient searches, any signal is lost in noise and attention is deployed at random from item to item. A simulation with this architecture was able to reproduce an extensive set of basic search results with a single set of parameters. In most respects the means and variances of the simulated RTs were similar to human data.

With this background, we can go on to discuss the achievements during the period funded by the AFOSR grant.

PROJECT ONE: The Development of the Guided Search 3.0 Model of Visual Search:

The Motivation for Guided Search 3.0

Like any model, GS2 made a series of simplifying and sometimes unrealistic assumptions. We have been working on the third generation of the model in order to simulate a larger set of results with a smaller set of assumptions. In this section, we will describe a series of issues that were not satisfactorily handled by GS2. These limitations of GS2 define goals for GS3.

1. The input to GS2 was a symbolic description of the stimulus (e.g. red, 90 deg at location X, Y). GS3 takes gray scale images as input.

2. In GS2, all locations in the visual field were equivalent. This is, of course, incorrect. Acuity and sensitivity decline with eccentricity. There are very substantial eccentricity effects in visual search (e.g. Carrasco, Evert, Change & Katz, 1995; Chun & Wolfe, 1996, Wolfe, O'Neill, and Bennett, 1997) All else being equal, RTs are shorter for items near fixation. We have shown that the eccentricity effect in search is not a simple by-product of the decline in acuity and sensitivity. There appears to be an attentional bias toward central items (Wolfe, O'Neill & Bennett, 1997). This is modeled in GS3.

3. In GS2, the eyes don't move. In the real world, even in most laboratory search tasks, they do. With the large, simple stimuli used in standard search tasks, RTs do not depend on the presence or absence of eye movements (Zelinsky & Sheinberg, 1997). Search in the real world requires eye movements to direct the high resolution fovea to possible targets. A first attempt to model eye movements and attentional deployments together is included in GS3.

4. In GS2, top-down and bottom-up activations are computed for various features and then combined into an attention-guiding activation map that remains constant for the duration of the trial. This is unrealistic, particularly if eyes are allowed to move. In GS3, attentional guidance can be updated during the course of a search.

5. Virtually all visual search work has been done with static observers viewing static stimuli. In the real world, observers move and objects move. As a consequence, the objects of search move in the image. Models of search should be able to deal with this situation. This is an area for future development.

Guided Search 3.0α

The current version for the Guided Search simulation should be called called "GS3α" since it is still in development. Nevertheless, it accomplishes many of the goals set forth above. Figure 6 shows the main components of GS3α. Each box shows an image that is an example of activity in a 2D array of 150x150 units, representing a 15 x 15° portion of the visual field. Each of the modules in GS3 has an analog in the physiology of the visual system.

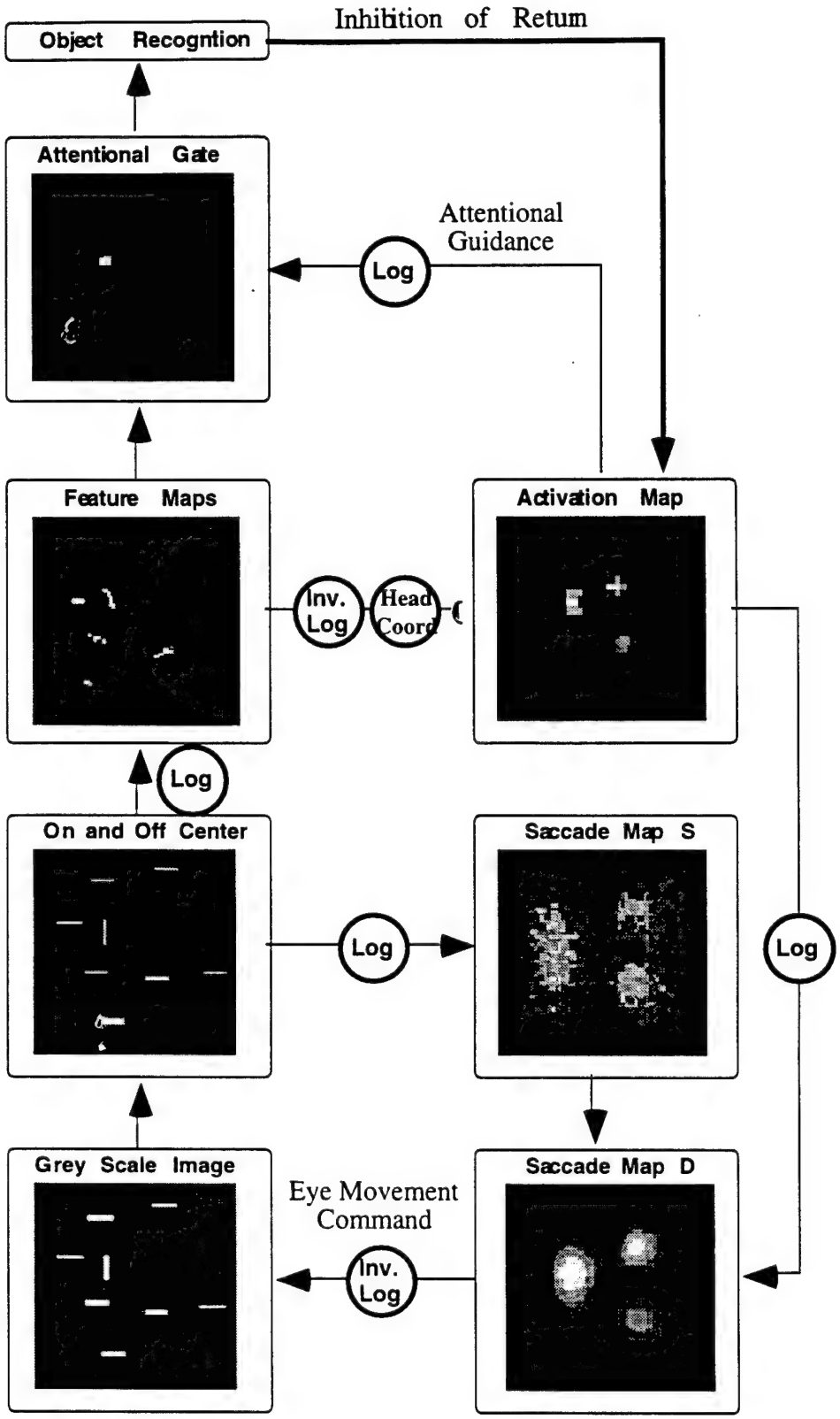


Figure 6 - Guided Search 3.0α

Grey Scale Image

The Grey Scale Image corresponds to the retinal image. Unlike GS2, GS3 determines the features of items on its own. In principle, this means that the model can be given arbitrary images as input. At present, however, only rather limited orientation and luminance preprocessing is implemented. Other aspects of early visual processing (color, motion, size, etc.,) would need to be added in order to achieve our eventual goal of biologically realistic operation on arbitrary images.

On and Off-Center Units

The next stage in the model consists of units approximating the circular center-surround receptive field organization of retinal ganglion cells. Each unit's receptive field consisted of a small excitatory central area and a broader inhibitory surround. The on-center responses are shown in the model diagram. In this case, light increment excites the center of the unit. Both on-center and off-center responses were computed in the simulation. The response of each unit was thresholded to reduce noise.

Preattentive Feature Maps

At present, GS3 implements preattentive processing only for orientation (Foster & Ward, 1991) and luminance polarity (Theeuwes & Kooi, 1994). Even those features are simplified versions of their counterparts in human vision. The output of the on and off-center units is fed into four feature maps coding for bright, dark, vertical, or horizontal local activity in the grey scale image. Bright detectors obtained their input directly from the on-cells, dark from the off-cells. Vertical and horizontal detectors gained their orientation preference by receiving input from spatially aligned on and off-units representing a line in the image, as suggested by Hubel and Weisel (1962).

All of the components of GS2 were in retinal coordinates. This is inadequate if we wish to model eccentricity effects and eye movements. By the time input reaches visual cortex, the central portions of the field are massively overrepresented. This cortical magnification of the foveal region can be approximated by a complex log transform (Schwartz, 1977; Schwartz, 1980). The effect of the transform is to split the field into two hemispheres and to magnify the area near fixation at the expense of more peripheral loci. These effects can be seen in the "feature map" box of Figure 6. Also note that the foveal region shows very little warping compared to the periphery[3]. For GS, the consequence of this mapping is that attention-guiding information is likely to be better near fixation than at more remote loci. Thus, target near fixation will attract attention more readily than peripheral targets.

The Attentional Gate

As noted earlier, an important role for attention is to restrict the flow of information from early, preattentive levels (feature maps) to later processing stages (in this case, object recognition). The attentional gate allows information from only one item to reach the object recognition module. The position of the opening in the gate is determined by the activation map (discussed below).

Object Recognition

[3] Feature maps were first computed in a Cartesian coordinate frame (that of the image, and on-center and off-center units), and then transformed into the log frame. To realize the full computational savings of the log map, feature map response would need to be computed directly in the log frame. However, to do so requires that the receptive field of units in each feature map vary based on the position of space they code for. Our technique avoided this complexity, though at greater computational cost.

In the present simulation, the endpoint is object recognition. It could be any limited-capacity process (e.g. spatial relationships - Logan, 1995). Our current object recognizer only identifies four types of items: black vertical, white vertical, black horizontal, and white horizontal. This is adequate to illustrate the basic functions of the model. Without attention, the object recognition module acts like an IT cortex cell in an anesthetized monkey with a receptive field covering the whole field (Gross, Rocha-Miranda & Bender, 1972; Schwartz et al., 1983). The attentional gate effectively shrinks the receptive field to one area or item of interest (Moran & Desimone, 1985a).

Activation Map

The attentional gate is under the control of an overall activation map. As in earlier versions of Guided Search, this activation map is a weighted sum of the outputs of the feature maps. For example, when looking for a bright and vertical target, the model would have a high weight connecting all bright feature detectors and all vertical feature detectors to units in the activation map. This top-down weighting is symbolized in Figure 6 by the filled-in half circle between the feature map and activation map.[4]

Activity at each spatial location in the activation map represents the best estimate of the preattentive processes that a target is located there. Attention is guided to the most active location in the activation map as determined by a winner-take-all computation. Specifically, this is implemented as a recurrent, on-center off-surround network (Grossberg, 1973). In this network, each item works to suppress other items until only the strongest remains active. In the simulation, this process takes about 50msec.

If the preattentive processes are incorrect and attention is directed to a distractor item, that item's activity in the activation map must be inhibited so that a new item can win in the network and attention can be deployed to the next most active item. In a model where the eyes are allowed to move, this "inhibition of return" (Posner & Cohen, 1984; Tipper et al., 1994) should be attached to the item and not to the retinal locus. The activation map, therefore, is computed in a head-centered coordinate system . Accordingly, the inputs from the feature map to the activation map undergo two transformations: Inverse-log and a translation based on eye position. The output of the activation map is log transformed in order to be consistent with the feature map output that it is gating. Inhibition of return is shown in Figure 6 as a feedback from the Object Recognition decision module to the activation map.

Saccade Maps

We are attempting to model eye movements during visual search with minimal additions to Guided Search. In GS, covert attention is deployed at a rate of about 20Hz. We know that saccades are made more slowly - once every 200 to 250 msec. Nevertheless, attention and saccades are intimately related. While it is possible to deploy attention without moving the eyes, it does not appear to be possible to make a saccade without deploying attention (Hoffman, 1996; Kowler, Anderson, Dosher & Blaser, 1995). In GS3, the saccade generator is simply a version of the activation map that is coarser in space and time. In space, the activation map is blurred to yield the saccade map. If the winner-take-all aspect of the activation map has not yet selected a sole winner, blurring can have the effect of averaging neighboring peaks of activation and, thus, permitting saccades to go to

[4] In GS3 top-down actiavtion is realized through the weighted connections from feature maps to the activation map. Bottom-up activation is a property of the feature maps. Thus, a single bright item would attract attention bottom-up because it would produce a peak of activation in the "bright" feature map which would be passed along to the activation map. If it is the case that some irrelevant singletons capture attention, that would imply that the weights linking the feature maps to the activation map can never be set to zero (discussed in Wolfe, 1994).

intermediate/compromise locations (Findlay, 1995; Zelinsky & Sheinberg, 1997). In the temporal domain, eye movements are generated by a clock every 200-250 msec while attention is deployed to the item with highest activation in the activation map approximately every 50 msec.

GS3 has two saccade maps intended to be analogous to the superficial and deep layers of the superior colliculus[5]. The two major projections of retinal ganglion cells are V1 and the superficial layers of the superior colliculus (Spillman & Werner, 1990). As a simplified model of this biology, we took the log transform of the on and off-center stage response and fed that into the saccade map superficial stage of the model. The log mapping has been shown to provide a good quantitative fit to the spatial mapping of the superior colliculus (Ottes, Van Gisbergen & Eggermont, 1986). A 5mm x 5mm region of the colliculus was modeled, using parameters found by Ottes et al. (1986). For the present, this pathway is included for completeness. It does not contribute to the simulation results.

The blurred activation map that actually generates eye movements in GS3 is a second saccade map modeled after the deep layers of the superior colliculus. When cells in the deep layers of the colliculus are electrically stimulated, each cell produces a saccade of a specific size and direction, regardless of the eye's initial position (Robinson, 1972; Sparks & Mays, 1980). In the GS3 model, the direction and magnitude of a saccade are represented by the site of highest activity in this second saccade map.

GS3 Performance

Visual Search Tasks - RT x Set Size Measures

The GS3 simulation has been tested on three standard search tasks similar to those illustrated in Figures 1 and 2. The Feature Search was a search for a dark target among bright distractors. The Conjunction Search was a search for a dark vertical target among dark horizontal and bright vertical distractors (Fig 2a). Finally, the serial search was logically equivalent to a search for a T among Ls (Fig 1b). Serial search was simulated by turning off top-down guidance. This allows attention to be deployed at random from item to item.

A 'yes' response was generated whenever the simulation's object recognition model decided that it found the target. A 'no' response was generated when no activations exceeded an activation threshold set dynamically in a manner similar to that described in Chun and Wolfe (1996). Set sizes of 5, 10 and 15 items were tested. 5000 trials were run divided evenly between target present and target absent conditions. Results are given in Figure Seven:

[5] The saccade-generation aspects of GS3α are similar in conception to the model of Sheinberg and Zelinsky (1993).

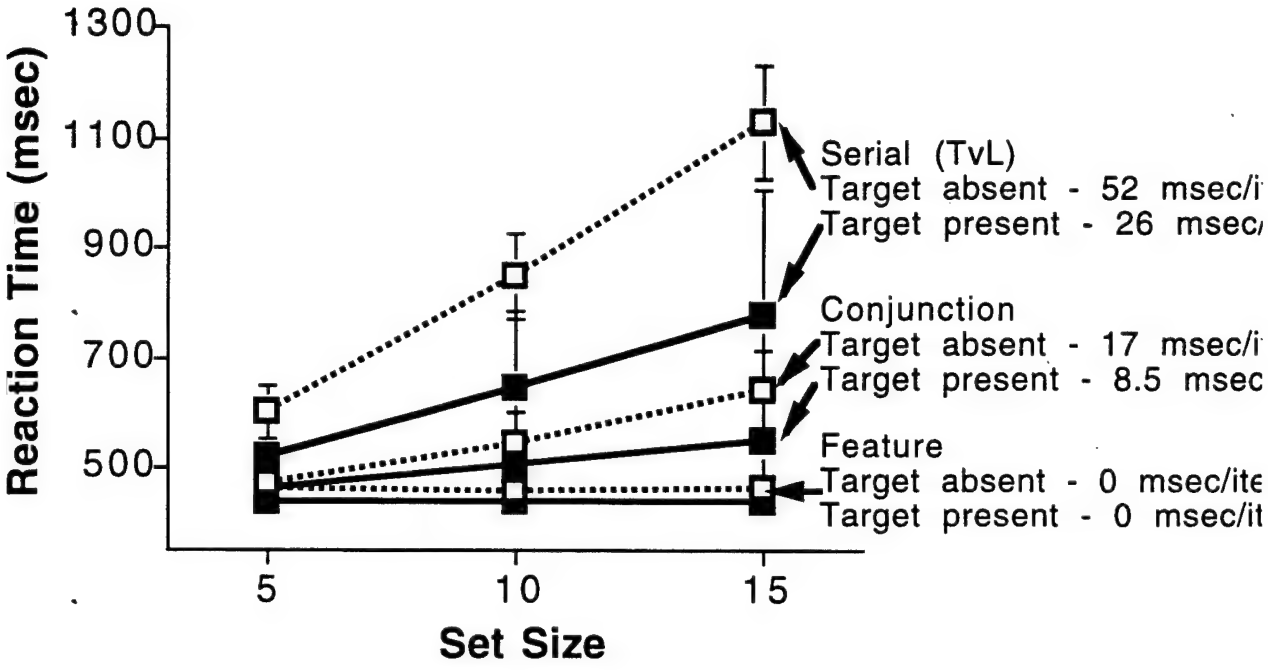


Figure Seven - RT x set size functions simulated by GS3

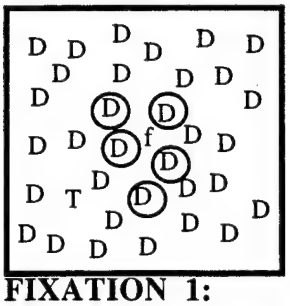
Slopes of the RT x Set Size functions are comparable to human results for these tasks (e.g. Wolfe et al., 1989). Error bars show +/- 1 standard deviation. As in the GS2 simulation, serial task, target absent variances are low compared to the human data. Error rates were 14% for the serial task, 13% for the conjunction task, and 8% for the feature task. These are somewhat higher than human performance. As in the human data (Wolfe et al., 1997), GS3 makes most of its errors with targets in peripheral locations. The high error rates may indicate that the simulation's peripheral resolution may be somewhat too coarse. Further aspects of the human and simulated eccentricity effects are discussed in a separate section, below.

Eye Movements

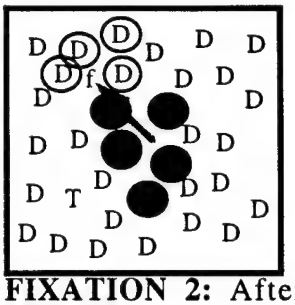
GS3 treats eye movements in search as slaves of attentional deployment. Attention proceeds from item to item approximately every 50msec. The eyes move every 200-250 msec with their destination based on a saccade map that is a blurred version of the activation map that directs attention. Data from Findlay (1995) can be modeled as a by-product of this blurring. Findlay presented two targets and instructed Ss to make a saccade as quickly as possible to one of them. He found a speed-accuracy trade-off. Short latency saccades tended to go to intermediate positions between the targets. Longer latency saccades went accurately to one of the two targets. GS3 shows the same behavior because at short latency, the winner-take-all network in the activation map will still have multiple peaks of activation. At longer latencies, there is only one peak and a blurred version of a single peak is still a single peak.

Eye movements in visual search

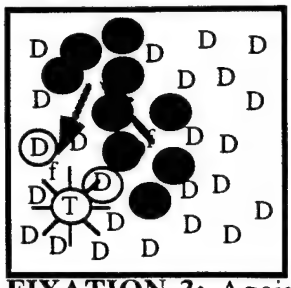
GS3 proposes a constructive interaction between eye movements and attentional deployments in visual search, as illustrated in Figure Seven. In this figure, "T" indicates the target, "D"s are distractors, and "f" is the point of fixation. In this illustration, we assume a fully serial search with no attentional guidance.



FIXATION 1: Fixation is at "f". Because the central items tend to have higher activation than peripheral items in GS3. The first items examined will be near fixation (indicated by open circles).



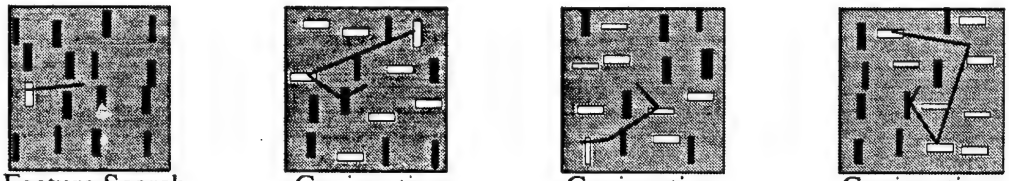
FIXATION 2: After 200-250 msec, the eyes move. Each attended distractor has been rejected and inhibited (filled circles) so the eyes move to a different portion of the display and attention is deployed to items near the new fixation.



FIXATION 3: Again, the attended items are distractors and are inhibited. A new saccade is generated and attention begins to explore the items around the new fixation. The target is found and search is terminated.

Figure Seven: Eye movements and attentional deployments in GS3

Note that the search time, defined as number of items attended times 50 msec plus some constant, would be the same in this example if the eyes were held fixed. Different items would be attended but, on average, 50% of items would need to be attended on target present trials. This is consistent with the findings of Zelinksy & Sheinberg (1997). However, this independence from eye movements only occurs if all items can be resolved in peripheral vision. In more realistic search tasks, eye movements and attentional deployments work together. The eyes direct the fovea to a region allowing covert attention to explore items in that region. Eye movement records will approximate a coarse sampling of the covert deployments of attention with one eye movement for every 4 or 5 attentional deployments. Sample eye movement scan paths from the GS3 simulation are shown below:



Feature Search Target Present | Conjunction Target Present | Conjunction Target Present | Conjunction Target Absent

Figure Eight - GS3 scan paths

GS3 - Summary

To summarize, Guided Search 3.0 builds on our successful modeling of laboratory-based visual search tasks. With GS3α, we have taken important steps toward a model of search behavior in natural scenes. Specifically, the introduction of eccentricity effects acknowledges that processing varies across the retinal image. As a consequence, attention will generally be best deployed at or near fixation. In a system with a fovea, the eyes must be allowed to move in order to take proper advantage of foveal processing. In GS3, we can move the eyes with essentially the same mechanism that we use to deploy attention. The result, in extended, complex scenes will be cooperation between a fast covert attention deployment and slower saccadic eye movements. Returning to the list of motivations for GS3 given in the Background section, we can see that GS3α represents a significant advance over GS2:

1) GS3α takes gray scale images as input.
2) It models eccentricity effects.
3) GS3α produces eye movements with minimal additions to the circuitry needed to deploy attention in search tasks.
4) GS3α updates the guidance of attention during the course of a single search.

PROJECT TWO: Eccentricity Effects in Visual Search:
In standard visual search experiments, observers search for a target item among distracting items. The location of target items is generally random within the display and ignored as a factor in data analysis. Most models of visual search, including the earlier versions of our Guided Search model, likewise ignore the effects of target location. The goal of this project was to eliminate this oversimplification. Previous work has shown that targets presented near fixation are, in fact, found more efficiently than targets presented at more peripheral locations (Carrasco et al, 1995; Geisler and Chou, 1995). Moreover, work with search-like tasks had shown that the "useful field of view" was related to performance on real-world tasks (e.g. driving (Ball, Owsley, Sloane, Roenker & Bruni, 1993). The useful field of view (UFOV) is a somewhat arbitrary visual field isopter. Inside the UFOV, a specific task (e.g. search) can be done. Outside the UFOV, subjects fail to perform the task to some criterion level. Ball and her colleagues report that the UFOV is smaller in older individuals than in the young (Ball, et al., 1988). It shrinks in the presence of cognitive load (Graves, et al., 1993) and correlates with automobile accidents (Ball, et al., 1993).

For our purposes, the existing data were not adequate. The UFOV data was collected with an unusual task and the search data were limited in extent and did not examine the effects of load. Therefore, we embarked on a series of experiments in which RTs on visual search tasks were examined as a function of target eccentricity.

Methods
These studies are variants of standard visual search tasks. Stimuli are presented on a standard Macintosh computer monitor in a region that subtends 16 by 16 deg. Set sizes of 25, 35, and 45 items are used. We use these relatively large set sizes so that, even in a relatively efficient conjunction search task (Wolfe, 1992), attention would need to be deployed to several items in each search. We also test subjects with a single item (set size 1). This is analogous to a standard visual test where there is no ambiguity about the location of the target. Set size 1 serves as a control condition.

The 16 x 16 deg screen is divided into a 7 x 7 array of cells. Distractors can appear in any cell except the cell at fixation. To reduce the numbers of trials per subject, targets are constrained to appear in the 24 locations forming 8 arms radiating from fixation (see Fig. Nine).

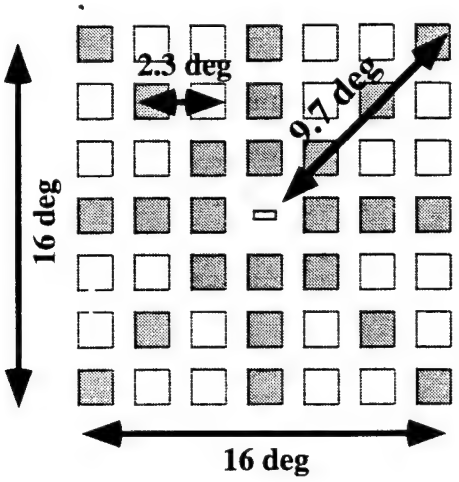


Figure Nine: Stimulus configuration for all experiments. Distractors could appear in any of the 48 loci (excluding fixation). Targets were constrained to appear only in the shaded loci. Neither boxes nor shading was present in the actual stimulus display.

Cutting the number of target locations in half saves hundreds of trials. Target locations are not shaded or otherwise indicated in the display seen by the subjects. Subjects will not know the distribution of target locations and, even if they did, there is no evidence that subjects can spread attention in the octopus shape required to exploit that information.

Within the constraints just described, placement of targets and distractors is random from trial to trial. Targets will be presented on 67% of the trials (Blank trials are required to keep the subject "honest" but are relatively uninteresting in a study of target location.) Before each session, the task will be described to the subject. Subjects will be asked to respond as quickly as possible, while making between 5 and 10% errors. They will not be informed about the ratio of target-present to target-absent trials. Subjects are tested for 30 practice trials followed by 300 experimental trials.

Results

Typical data from ten subjects is shown in Figure Ten.

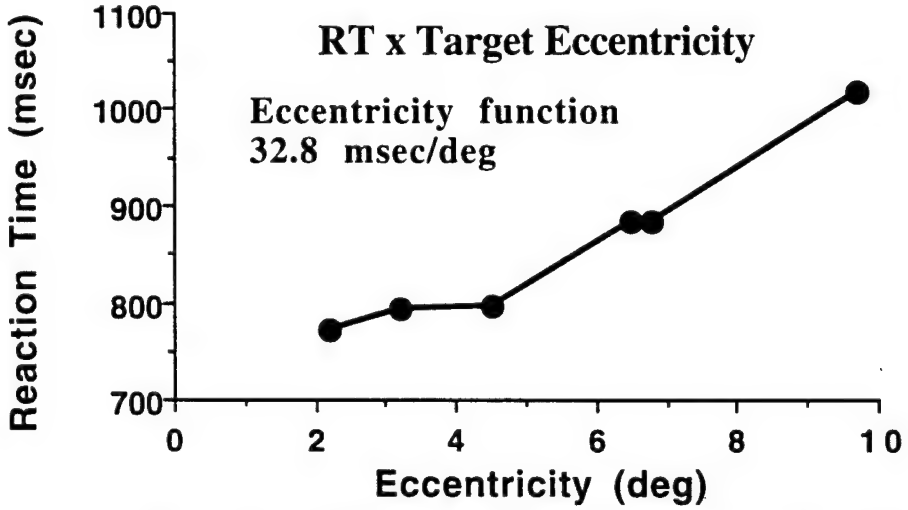


Figure Ten: RT as a function of target eccentricity

Wolfe, O'Neill, and Bennett (1997) is the archival report on a series of experiments intended to determine the cause of the eccentricity effect in visual search. In brief, the paper proposes that the primary cause is an attentional bias that allocates attention preferentially to

central items. The first four experiments deal with the possibility that "front-end" visual, and not attentional, factors underlie the eccentricity effect. They show that the eccentricity effect shown here cannot be accounted for by the peripheral reduction in visual sensitivity, peripheral crowding, or cortical magnification. Certainly, visual factors can produce an eccentricity effect directly (e.g. Geisler and Chou, 1995). In our experiments, with large, well-spaced stimuli, however, the eccentricity effect is a matter of a bias in attentional deployement.

GS3 models eccentricity effects by distorting the input in accord with a cortical magnification factor. This causes activations of otherwise equivalent stimuli to be greater for stimuli that are closer to fixation. Higher activation increases the chances that attention will be deployed to that item.

Experiment Five in Wolfe, O'Neill, and Bennett (1997) is intended to be a test of the attention allocation model. It also shows that RT x set size effects can be independent of eccentricity effects. This is important since most models of visual search rely on the assumption that RT x set size functions are not a simple artifact of RT x eccentricity functions. If subjects search from fixation outward, then the effective set size in a search task is dependent on the eccentricity of the target. That is, distractors that are more peripheral than the target are unlikely to attract attention. Experiment Six is designed to look for this effect and to further examine the relationship of RT x set size functions to target eccentricity. In standard search experiments, this relationship does not become a problem unless target location is constrained in a way that makes the average target eccentricity systematically different from the average distractor eccentricity.

GS3 and Eccentricity Effects

Our experiments show that eccentricity effects vary with the efficiency of the search task. Thus, if you are searching for a T among L's, you will find a "T" near fixation much sooner than if it is tucked away in the corner of the display. If the task is easier, say, a search for a conjunction of color and orientation, the effect of eccentricity will be less marked. If there is just a single item on the screen, attention will be deployed to it without impediment and there will be minimal effect of eccentricity. These basic results are shown in Figure Eleven for both human subjects and for the GS3 simulation.

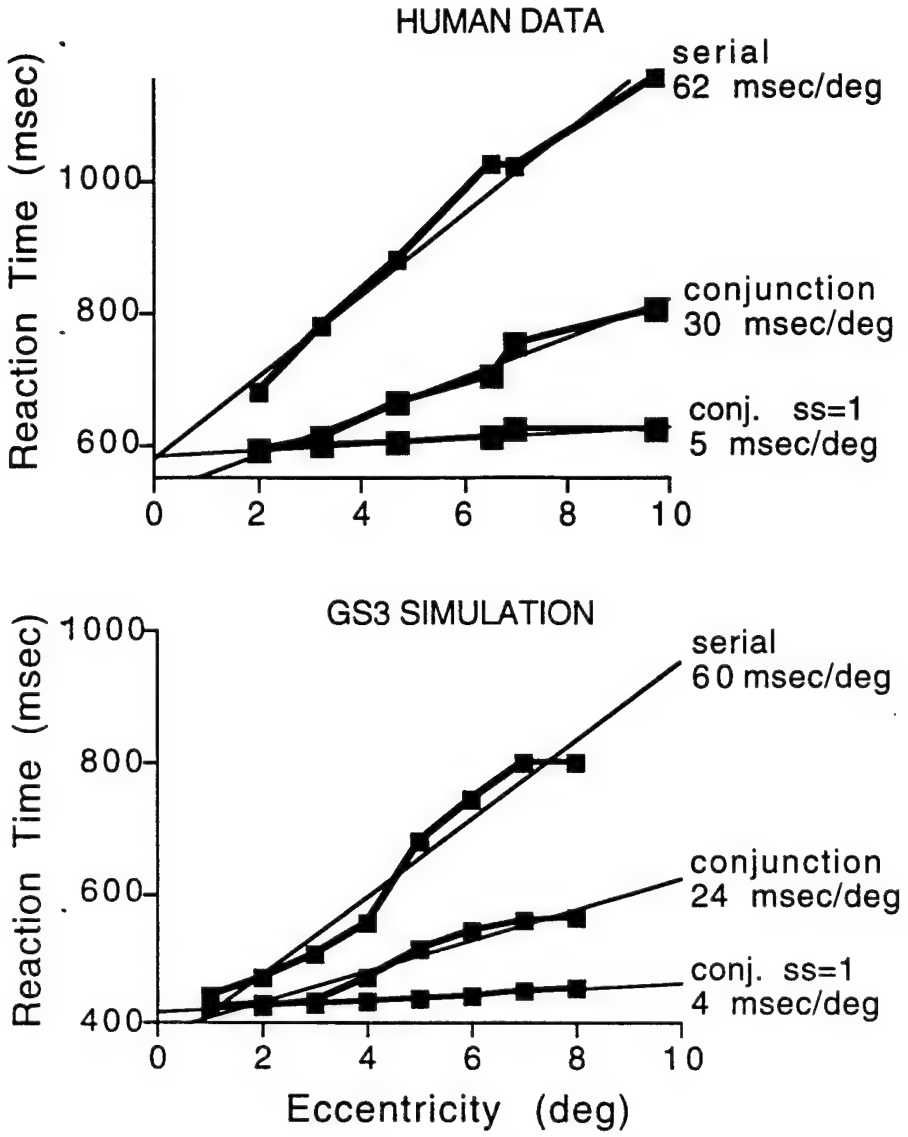


Figure Eleven - Eccentricity effects in visual search: Human and GS3 data

The simulation provides a good fit to the human data, showing that we can account for eccentricity effects with a small and physiologically plausible modification of Guided Search.

Effects of Load

To study the effects of load on the eccentricity effect, subjects are asked to perform two tasks at the same time. The first is a conjunction search task as described above. The second is a vigilance task. The display is identical to that shown in Figure Nine. For the vigilance component of the task, subjects monitor the rectangle at fixation. If it turns from white to red, they make a speeded response. While they are monitoring the vigilance target, the search stimuli can appear. Subjects are instructed to find the red vertical target but always to give priority to responding to the vigilance target if it changes color. In the control condition, subjects make the visual search response, ignoring the vigilance task. Results for a population of normally-sighted young subjects are shown in Figure Twelve, below:

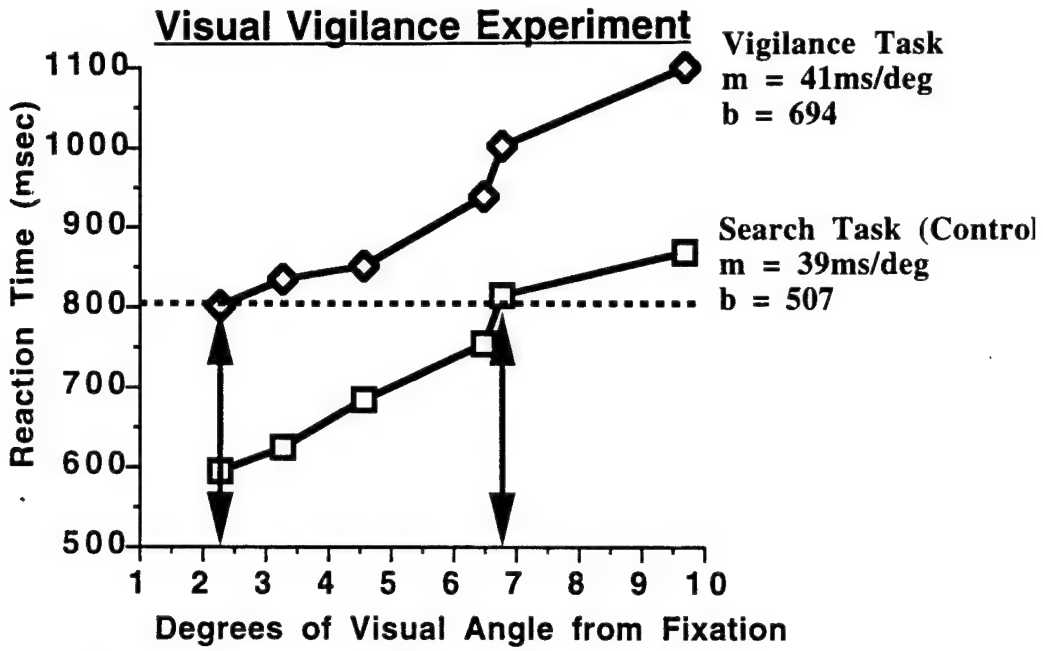


Figure Twelve - Effects of adding a vigilance task to a visual search task

Since the subjects had instructions to give priority to the vigilance task, responses on that task were fast and accurate. The addition of the vigilance task had a large, additive effect on the eccentricity effect in visual search. No matter where the visual search target was located, the RT for finding that target was 200 msec longer when subjects were performing a concurrent vigilance task. To see how this translates into a shrinking of the useful field of view, consider a task that must be accomplished in 800 msec (shown by the dashed line). For the standard search task (the control condition), a target can be found within 800 msec if it is not more than about 7 deg eccentric. Adding the vigilance task moves that 800 msec threshold to just 2 deg of eccentricity. Note that shrinkage of the UFOV could be accomplished by an increase in the slope of the eccentricity function. This would occur of the added load slowed each deployment of attention. Instead, in several versions of this experiment, we see an additive effect, suggesting that subjects were dividing their time between vigilance and search tasks.

We have run several variants of this task. If the vigilance task is made more difficult by making the change in the fixation stimulus more subtle, the additive cost of vigilance increases. We changed the vigilance task so that the entire background of the display changed color. A significant cost of the vigilance task remained. This shows that the cost of vigilance was not due to the need to attend to two spatial locations at the same time (vigilance at the center, search elesewhere). In this experiment, the vigilance stimulus was present wherever attention was deployed. We changed the vigilance task to an auditory monitoring task. In this case we found no cost of the added task. Though it is diffiuclt to compare across sensory dimensions, auditory and visual vigilance tasks were matched for RT and accuracy. This raises the interesting possibility that the auditory and visual tasks draw on different pools of attentional resources. Further experiments are needed in this area. In a related experiment, we used a memory task in place of the vigilance task. Subjects were required to hold 1, 3, 5, or 7 numbers in memory while performing the visual search task. There was no cost of this added load. In this case, it would be worthwhile to increase the cognitive demands of the task (e.g. add the numbers). At some

point, one imagines, cognitive load must impair performance on the search task. A simple memory load, however, was not sufficient to interfere with visual search.

Effects of Visual Degradation

Another source of visual load or stress is the nature of the visual input itself. Specifically, we wondered if an imbalance between the inputs to the two eyes would interfere with search. Blur of one eye is naturally occuring problem (anisometropia). It is also an artificial solution in situations where it is deemed useful to have each eye focused for a different distance (monovision)(Schor & Erickson, 1988). We tested subjects on the basic visual search - eccentricity effect paradigm in six conditions:

Condition	Description	Reason
0D/0D	no blur	standard control condition
4D/4D	4 diopters (D) blur in each eye	Does massive blur of the image disrupt search?
4D/2D	4D in one eye, 2D in the other yields 2D of anisometropia	Does mismatch between the eyes disrupt search.
2D/closed	2D blur in one eye, other eye closed	A control to determine if 4D/2D is equivalent to monoc. 2D
0D/0D set size 1	0D/0D with a single item on screen	Control for the visibility of stimuli
4D/2D set size 1	4D/2D with a single item on screen	Control for the visibility of degraded stimuli

4D/2D was used for the anisometropic case because less extreme conditions (0D/2D, for example) allow subjects to place accommodation in an intermediate state, equalizing blur in the two eyes. With 4D/2D, this is impossible. Large stimuli were used because it is uninteresting to show that search is impaired if the stimuli are blurred to the point of unrecognizability. Results are shown in Figure Thirteen, below:

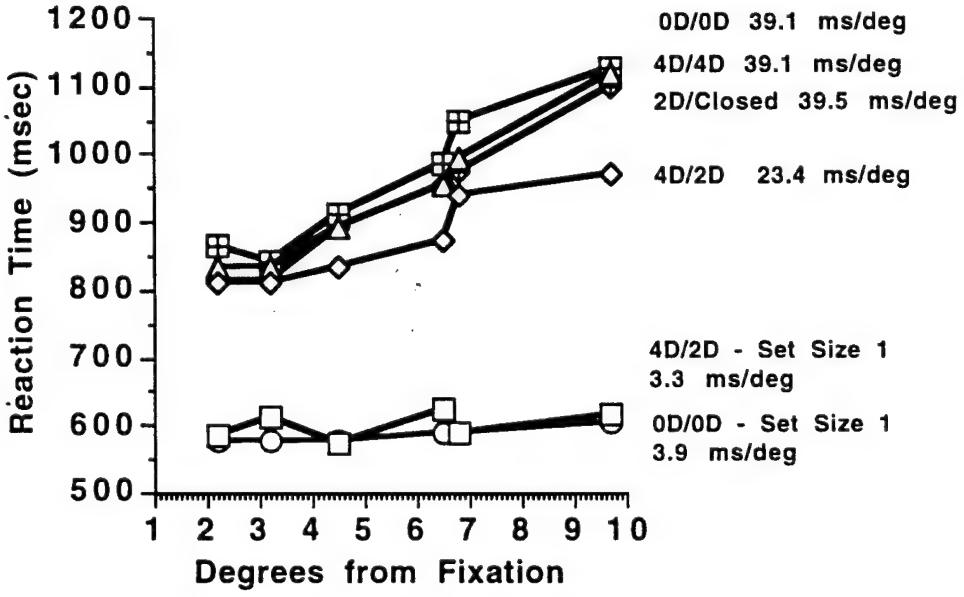


Figure Thirteen: Effects (or lack of effects) of visual degradation on eccentricity effects in visual search

Looking first at the lower two curves for set size 1, there is no effect of visual degradation. This shows that the stimuli could be identified in isolation at all positions in the display. Turning to the four upper curves, the only effect of visual degradation would seem to be an improvement in performance for the most taxing 4D/2D condition. This difference does not

appear to be a speed-accuracy trade-off. It may reflect additional effort on the part of the subjects in this condition, but one would want to replicate the result before drawing any firm conclusions. We can say that there is no evidence that these forms of visual degradation and binocular imbalance interfere with visual search.

Summary:
The work on eccentricity effects shows that attentional deployment is modulated by eccentricity. GS3 can model that effect. Added visual loads produce an additive effect on search RTs in these tasks. Auditory and memory loads had no effect. Likewise, visual degradation and binocular imbalance had no effect in these cases.

PROJECT THREE: What 1,000,000 Trials Tell Us About Visual Search
In a typical visual search experiment, RT is measured as a function of set size. Inferences about the underlying search processes are based on the slopes of the resulting RT x set size functions. These are the data for informal and formal modeling of visual search. A problem arises, however, when assumptions about the nature of a specific type of search task are based on the results of one or two experiments. Most search experiments involve 5-15 subjects performing a few hundred trials each. These results can tell you something about the mean RT x set size functions, but much less about the distributions of those functions. It is not neccessary to run thousands of subjects on thousands of trials. Instead, we have pooled results from many experiments, run over the last decade. In this retrospective study, we examine results from 2500 subjects performing a few hundred searches each in a wide variety of search tasks (approx. 1,000,000 search trials). These large numbers permit us to see statistical properties not visible in smaller data sets. The details are described in Wolfe (1997). Here we describe some highlights:

1) It is popular in the search literature to assume that search tasks can be categorized as "serial" or "parallel" on the basis of RT x set size slopes. However, as shown in Figure 14, the distribution of search slopes is unimodal with no evidence of bimodality.

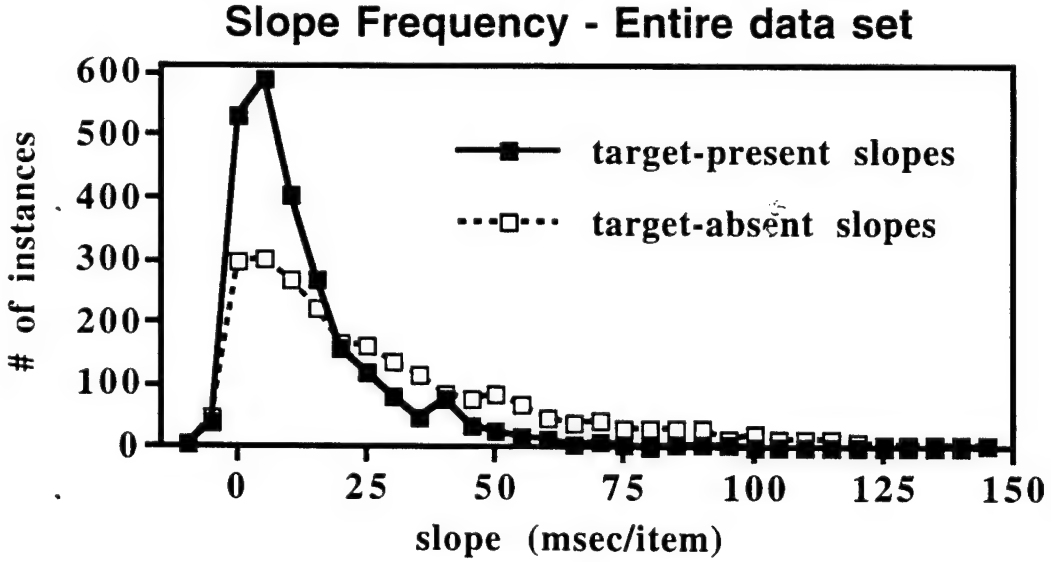


Figure Fourteen: Distribution of 2500 RT x set size slopes

2) A ratio of 2:1 is considered to be a diagnostic of a serial self-terminating search. A ratio of 1:1 is sometimes considered to be a diagnostic of a parallel, unlimited-capacity search. In this data set, the mean and median ratios of target absent to target present slopes are significantly greater than 2:1.

3) As shown in Figure 15, different classes of search task (e.g. feature, conjunction, spatial configuration) produce different but overlapping distributions of slopes.

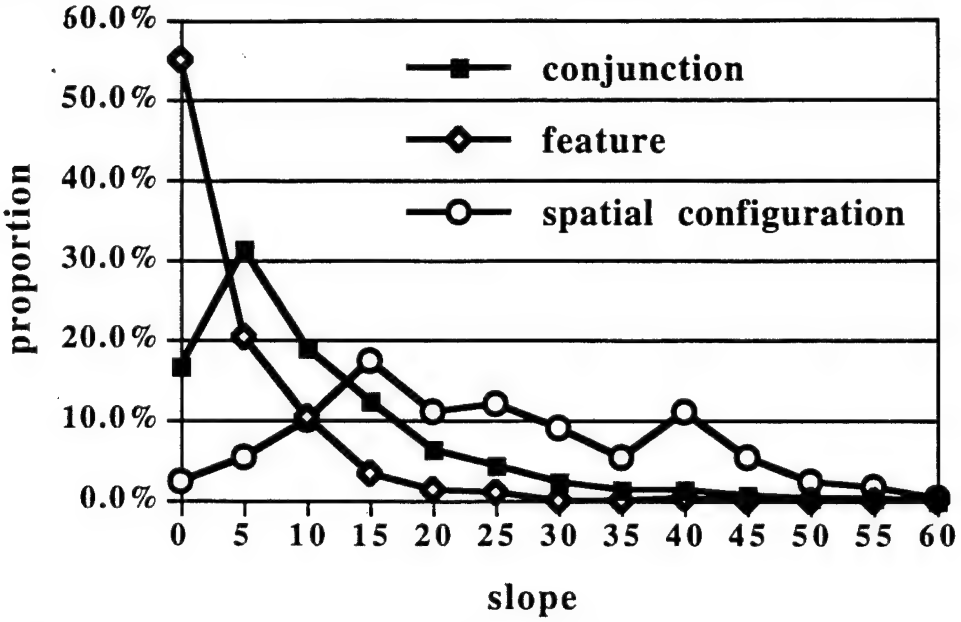


Figure Fifteen: Distribution of target-present search slopes for feature, conjunction, and spatial configuration (e.g. T vs L) searches.

Note that any of these search tasks might produce a slope of moderate efficiency (e.g. 10 msec/item). Thus, while the distributions are significantly different, it will be risky to categorize search tasks on the basis of a few slopes.

4) As shown in Figure 16, the different types of search task differ in the ratios of target absent to target present slopes.

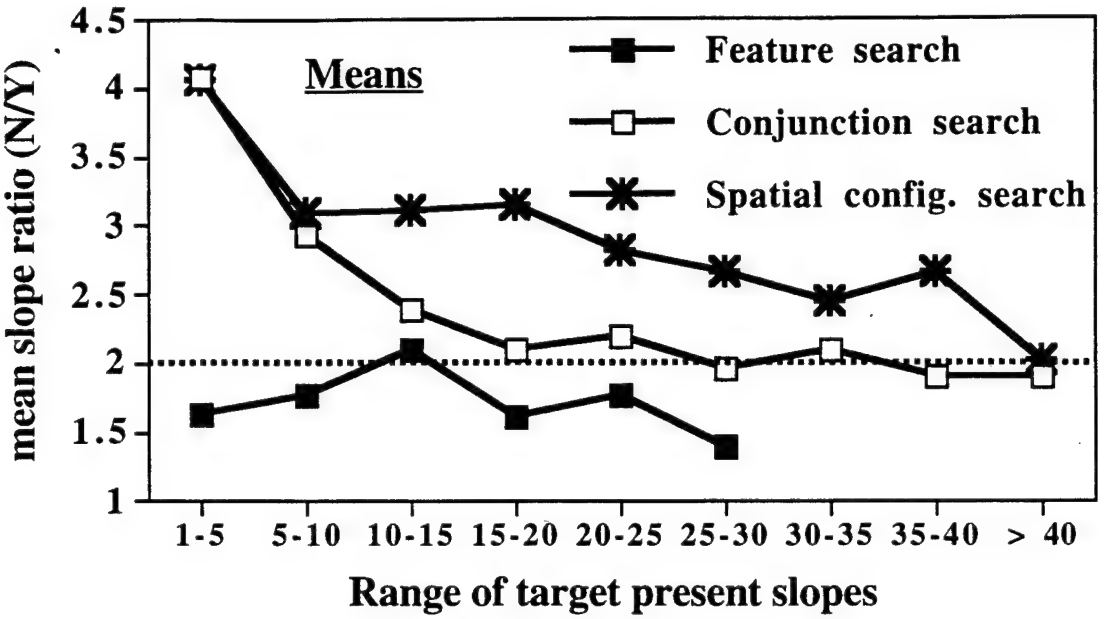


Figure Sixteen:
Slope ratios as a function of search task and target present slope.

Several points of interest are illustrated by this graph:

1) Spatial configuration searches are the gold standard "serial" searches: Ts among Ls and 2s among 5s. As such, they would be expected to produce slope ratios of 2.0. In fact, the slope ratios for these searches are significantly greater than 2.0 over almost the entire range of target present slopes.

2) Even the most efficient feature searches do not produce 1:1 slope ratios diagnostic of "parallel" search. Feature searches produce slope ratios significantly grater than 1.0.

3) Searches of the same efficiency need not be equivalent to one another. Consider the 5-10 msec/item bin. The conjunction, feature, and spatial configuration targets in this bin were all found with approximately equal ease. However, the process of terminating blank trials in feature searches is much more efficient than the process of terminating other types of searches as witnessed by the difference between the slope ratios.

These and other findings from the analysis of this large data set will constrain the future development of Guided Search. We will make the entire data set available to others on our website: www.dahlen.com/kari/wolfe.html.

PROJECT FOUR: Individual Differences in Visual Search

In most visual search experiments, the difference between results for individual subjects are treated as noise. That is, we behave as if 300 trials from subject A are the same as 300 trials from subject B. In this view, the only reason to test multiple subjects is that it would be too boring to inflict 3000 trials on a single subject. There is, however, another reason to run 300 trials on ten subjects rather than 3000 trials on one subject. That is the suspicion that all subjects are not the same and that data from multiple subjects are needed in order to paint a portrait of a 'typical' subject. Thus, differences between results of individual subjects could be treated as pure noise or could be seen as reflective of real differences in visual search.

The decision requires some study of those differences. This project is the start of such an investigation.

Two specific questions are addressed:

1) Are the individual differences seen in the RT x set size slopes of standard visual search experiments reliable?

2) Do reliable individual differences in visual search performance exist?

The answer to the first question is "no" but the answer to the second is "yes".

Experiment One: Reliability in standard visual search.

A standard psychometric approach to the question of reliability is to compare the results from one half of the data to the results from the other half. Thus, in a search experiment, if an RT X set size slope is derived from the even-numbered trials and another is derived from the odd-numbered trials, a reliable test should produce very similar slopes. In Experiment One, this so-called "split-half reliability" was used to test individual differences in standard search experiments.

Method

Participants. The same group of 30 participants participated in all 8 conditions. The participants were recruited from a pool of participants from Harvard Medical School, Northeastern University, and Massachusetts College of the Arts. All had normal or corrected to normal acuity, and all were tested for color blindness with Dvorine plates. Participants participated in two sessions, each lasting about 2 hours. There were four search conditions in each session. The order in which a participant ran the eight search conditions was pseudorandom. All participants gave informed consent for their participation and all were paid for their time.

Procedure and apparatus. As in our usual experiments, there were 330 trials in each condition; the first 30 were discarded as practice. Set sizes of 4, 10 and 16 items were used for all conditions. Participants were provided feedback after each trial and encouraged to keep error rates below 10%. There were three types of feature searches: color, size, and orientation, three conjunction searches, created by the pairwise comparison of each of the features and 2 spatial configuration searches using stimuli that were 2's and 5's drawn like the characters on a digital clock so that the 2 and 5 were mirror images of each other. In one condition, the target was the 2 with 5's as distractors. In the other condition, all stimuli were rotated 90 deg. The 2 and 5 stimuli were black, and subtended 1.83 by .98 degrees. For the size feature search, the target was a blue vertical line (CIE x, y coordinates: blue = .146, .065) subtending 2.4 by .5 deg among blue vertical lines subtending 1.2 by .5 degrees. Items that were 2.4 by .5 degrees of visual angle will be described as large, and items 1.2 by .5 degrees as small. For the color feature search, the target was the same blue vertical line among red vertical lines (all subtending 2.4 by .5 deg). For the orientation feature search, the target was a blue line tilted 45 degrees off vertical, among blue vertical lines (all 2.4 by .5 deg). For the Color-Orientation conjunction task, the target was a large blue line tilted 45 degrees off vertical, and the distractors were a large blue vertical line and a large red line (CIE x, y coordinates: red = .59, .33) tilted 45 degrees off vertical. For the Color-Size conjunction task, the target was a large blue line, and the distractors were a large red line (same dimensions) and a small blue line. All the lines were vertical. For the Size-Orientation conjunction task, the target was a large blue line tilted 45 degrees, and the distractors a small blue line tilted 45 degrees and a large vertical blue line.

The methods used here are comparable to those in many published search experiments (e.g. Treisman & Gelade, 1980; Wolfe et al., 1989). However, there are two major departures from standard practice that make this experiment unusual. The sample size of 30 subjects is much larger than the usual cohort which typically is less than 10. Secondly, all 30 subjects participated in all eight different tasks. The resulting dataset of 72,000 trials provides a degree of statistical power unavailable to most other studies in this field.

Results.

Group results. The group results will be presented first, to demonstrate that the stimuli and set sizes used in this experiment elicited standard search results. The slopes, averaged over 30 subjects are given below:

TASK	Target-present slope	Target-absent slope
Color Feature	1.1	-.7
Orientation Feature	.2	.4
Size Feature	3.0	4.8
ColorXSize Conjunction	5.5	11.8
Size X Orient. Conjunction	9.1	19.4
Color X Orient Conjunction	10.9	28.3
Upright 2 vs 5	21.8	56.4
Rotated 2 vs 5	44.4	89.4

These are in line with results from similar experiments in the literature.

Individual Differences

Reliability. For each condition one slope was derived from the even-numbered trials and another one from the odd-numbered trials. These were then correlated to provide split-half reliability for each condition (Carmines & Zeller, 1979; Nunnally, 1978) The reliability coefficients for all conditions are given in the table below:

Task	Target Condition	Slope Reliability	Mean RT Reliability
Color Feature	Present	.17	.94
	Absent	-.47	.96
Orient Feature	Present	.11	.98
	Absent	.59	.97
Size Feature	Present	.44	.97
	Absent	.59	.97
Color-Orientation	Present	.23	.98
	Absent	.66	.96
Color-Size	Present	.09	.97
	Absent	.87	.99
Size-Orientation	Present	.33	.97
	Absent	.83	.98
Rotated 2s	Present	.20	.92
	Absent	.87	.94
Upright 2s	Present	.58	.96
	Absent	.74	.98

The reliability coefficient is a correlation coefficient that ranges from 0 to 1, with 0 indicating no reliability, and 1 perfect reliability. Inspection of the table indicates that the obtained slopes from most of these conditions are fairly unreliable. In many cases, one would be hard pressed to find any systematic relationship between the slopes derived from the even numbered trials and those derived from the odd-numbered trials. Put into predictive terms, a reliability coefficient this low indicates that if one were using the slope derived from one half of the trials to predict the slope derived from the other half, percentage of explained variance would be 0.79% (on a 0 to 100% scale!). Reliability seems to be worst for the feature searches and best for the 2 vs. 5 conditions. Furthermore, reliability is higher, in general, for target absent trials. In fact, the target absent conditions for both 2 vs 5 searches are extremely reliable: r_{xx} for Upright 2 search, blank trials = .74, and r_{xx} for Rotated 2 search, blank trials = .87.

Given that the slope estimates for the target present trials for most of the tasks are quite unreliable, it might be useful to examine whether or not the underlying mean reaction times are reliable. The reliability of the mean reaction times might suggest reasons for the low reliability of the slope estimates. Low mean RT reliability would suggest that the data were inherently noisy. The motor component of reaction times could be variable, and swamp the search component. In any event, a lack of slope reliability would be unsurprising in the light of a lack of mean RT reliability. On the other hand, if the mean RTs are reliable, that would mean that the data points constituting the slope estimate are stable. In this case, we would infer that the variability is an artifact of the process of

deriving the slope estimate. In order to examine this, a mean RT was obtained from the even-numbered trials, and a mean RT from the odd-numbered trials, similar to the procedure used for the slope estimates. These were then correlated. The table above presents the mean RT reliabilities for all conditions in the fourth column. Reliability of mean reaction times is very high. Apparently, the unreliable slope estimates are based on reliable RT data.

Experiment Two: Are individual differences in visual search ever reliable?

The second experiment was similar to the first with the following important changes. Only two tasks were used: a color X orientation conjunction search and a 2 vs 5 search. In an effort to make the slope estimates more reliable, the number of set sizes was increased to four and the number of trials was increased to 1500 (1000 target present, 500 target absent). In addition, the eccentricity of the target was recorded (see Project Two, above).

With these data, we could examine reliability of the RT x set size slopes as a function of set size and number of trials. The reliabilities are shown in the table below.

	200 trials	**1000 trials**
set size 2	**r = 0.19**	**r = 0.61**
set size 4	**r = 0.54**	**r = 0.87**

These results show that the unreliability of the slope estimates in the first experiment can be overcome with a sufficient number of trials and set sizes. Why are the individual differences in standard search experiments unreliable? An answer comes from analysis of the eccentricity data. As discussed in Project Two, RT depends on the eccentricity of the target. In standard search tasks, each possible target location is represented only a very few times. Thus, if statistical fluctuation yields more eccentric targets at one set size than at another, the resulting RT x set size slope will be distorted. The effects of such random variation are reduced as the number of trials increases. Alternatively, one could restrict target locations to a circle of fixed eccentricity. From the present experiment, we can extract a subset of 200 trials at two set sizes all with targets at one eccentricity. Normally, slope estimates based on so few trials are unreliable. However, when targets are all of one eccentricity, the reliability is good (r= 0.82 in this case).

Discussion
Several conclusions can be drawn from this project.

1) Differences between slopes of individual subjects in standard search experiments are probably unreliable and should be regarded as noise.

2) Average slopes derived from multiple subjects are likely to be valid. The mean data from 30 subjects in Exp. 1 are very similar to the mean data from comparable experiments with ten subjects.

3) Experiments reporting data from a few subjects running a few hundred trials should be regarded warily.

4) Systematic studies of individual differences in visual search can be run but they will require thousands of trials per subject or restriction of target location to one eccentricity.

5) These studies, while they might be painful to conduct, may useful in future modeling of visual search since the pattern of individual differences across tasks may reveal modules in the underlying visual search "hardware".

6) Those interested in applied uses of visual search might want to develop more efficient tests of individual differences.

REFERENCES

Bacon, W. F., & Egeth, H. E. (1994). Overriding stimulus-driven attentional capture. Perception and Psychophysics, 55(5), 485-496.

Ball, K., Owsley, C., Sloane, M. E., Roenker, D. L., & Bruni, J. R. (1993). Visual attention problems as a predictor of vehicle crashes among older drivers. Investigative Ophthalmology and Visual Science, 34(11), 3110-3123.

Bauer, B., Jolicœur, P., & Cowan, W. B. (1996). Visual search for colour targets that are or are not linearly-separable from distractors. Vision Research, 36(10), 1439-1466.

Baylis, G. C. (1994). Visual attention and objects: Two object cost with equal convexity. J. Experimental Psychology: Human Perception and Performance, 20(1), 208-212.

Baylis, G. C., & Driver, J. (1993). Visual attention and objects: Evidence for hierarchical coding of location. J. Exp. Psychol. - Human Perception and Performance, 19(3), 451-470.

Behrmann, M., & Tipper, S. P. (1994). Object-based attentional mechanisms: Evidence from patients with unilateral neglect. In C. Umilta & M. Moscovitch (Eds.), Attention and performance 15: Conscious and nonconscious information processing., (Vol. 15, pp. 351-375). Cambridge, MA: MIT Press.

Bennett, S. C., & Wolfe, J. M. (1996). Serial search can proceed at 50 msec per item. Investigative Opthalmology and Visual Science., 37(3), S298.

Bilsky, A. A., & Wolfe, J. M. (1995). Part-whole information is useful in size X size but not in orientation X orientation conjunction searches. Perception and Psychophysics, 57(6), 749-760.

Carmines, E., & Zeller, R. (1979). Reliability and validity assessment. London: Sage Publications.

Carrasco, M., Evert, D. L., Change, I., & Katz, S. M. (1995). The eccentricity effect: Target eccentricity affects performance on conjunction searches. Perception and Psychophysics, 57(8), 1241-1261.

Chelazzi, L., Miller, E. K., Duncan, J., & Desimone, R. (1993). Neural basis for visual search in inferior temporal cortex. Nature, 363(27 May 1993), 345-347.

Chun, M. M., & Wolfe, J. M. (1996). Just say no: How are visual searches terminated when there is no target present? Cognitive Psychology, in press.

Cohen, A., & Ivry, R. B. (1991). Density effects in conjunction search: Evidence for coarse location mechanism of feature integration. J. Exp. Psychol. Human Perception and Performance, 17(4), 891-901.

Cowey, A. (1993). Seeing the tree for the woods. Nature, 363(27 May 1993), 298.

Dehaene, S. (1989). Discriminability and dimensionality effcts in visual search for featural conjunctions: a functional pop-out. Perception & Psychophysics, 46(1), 72-80.

Desimone, R., & Duncan, J. (1995). Neural mechanisms of selective visual attention. Annual Review Neuroscience, 18, 193-222.

Dick, M., Ullman, S., & Sagi, D. (1987). Parallel and serial processes in motion detection. Science, 237, p400-402.

Driver, J. (1992). Motion coherence and conjunction search: Implications for guided search theory. Perception & Psychophysics, 51(1), 79-85.

Duncan, J., & Humphreys, G. W. (1989). Visual search and stimulus similarity. Psychological Review, 96, 433-458.

Duncan, J., Ward, R., & Shapiro, K. (1994). Direct measurement of attention dwell time in human vision. Nature, 369(26 May), 313-314.

Egeth, H. E., Virzi, R. A., & Garbart, H. (1984). Searching for conjunctively defined targets. J. Exp. Psychol: Human Perception and Performance, 10, 32-39.

Findlay, J. M. (1995). Visual search: eye movements and peripheral vision. Optometry and Vision Science, 72, 461-466.

Foster, D. H., & Ward, P. A. (1991). Asymmetries in oriented-line detection indicate two orthogonal filters in early vision. Proceedings of the Royal Society (London B), 243, 75-81.

Friedman-Hill, S. R., & Wolfe, J. M. (1995). Second-order parallel processing:

Geisler, W. S., & Chou, K.-L. (1995). Separation of low-level and high-level factors in complex tasks: Visual Search. Psychological Review, 102(2), 356-378.

Graves, M. A., Ball, K. K., Cissell, G. M., West, R. E., Whorely, K., & Edwards, J. D. (1993). Auditory distraction results in functional visual impairment for some older drivers. Investigative Ophthalmology and Visual Science, 34(4), 1418.

Gross, C. G., Bender, D. B., & Gerstein, G. L. (1979). Activity of inferior temporal neurons in behaving monkeys. Neuropsychol., 17, 215-229.

Gross, C. G., Rocha-Miranda, C. E., & Bender, D. B. (1972). Visual properties of neurons in inferotemporal cortex of the macaque. Journal of Neurophysiology, 35, 96-111.

Grossberg, S. (1973). Contour enhancement, short term memory, and constancies in reverberating neural networks. Studies in Applied Mathematics, 52, 213-257.

Hoffman, J. E. (1979). A two-stage model of visual search. Perception and Psychophysics, 25, 319-327.

Hoffman, J. E. (1996). Visual attention and eye movements. In H. Pashler (Ed.), Attention, . London: University College London Press.

Horowitz, T. S., & Wolfe, J. M. (1996). Targets and anti-targets: excitation and inhibition in the guidance of visual search for conjunctions. Investigative Opthalmology and Visual Science., 37(3), S298.

Hubel, D., & Weisel, T. N. (1962). Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. J. Physiol., 160, p106-14.

Julesz, B. (1986). Texton gradients: The texton theory revisited. Biol. Cybern., 54, 245-251.

Julesz, B., & Bergen, J. R. (1983). Textons, the fundamental elements in preattentive vision and perceptions of textures. Bell Sys. Tech. J., 62, 1619- 1646.

Kim, M., & Cave, K. R. (1995). Spatial attention in visual search for features and feature conjunctions. Psychological Science, 6(6), 376-380.

Kowler, E., Anderson, E., Dosher, B., & Blaser, E. (1995). The role of attention in the programming of saccades. Vision Research, 35(13), 1897-1916.

Kwak, H., Dagenbach, D., & Egeth, H. (1991). Further evidence for a time-independent shift of the focus of attention. Perception & Psychophysics, 49(5), 473-480.

Logan, G. (1995). Linguistic and conceptual control of visual spatial attention. Cognitive Psychology, 28, 103-174.

Mack, A., Tang, B., Tuma, R., & Kahn, S. (1992). Perceptual organization and attention. Cognitive Psychology, 24, 475-501.

Moraglia, G. (1989). Display organization and the detection of horizontal line segments. Perception & Psychophysics, 45(3), 265-272.

Moran, J., & Desimone, R. (1985a). Selective attention gates visual processing in the extrastriate cortex. Science, 229, 782-784

Nagy, A. L., & Sanchez, R. R. (1990). Critical color differences determined with a visual search task. J. Opt. Soc. Am. - A, 7(7), 1209-1217.

Nakayama, K., & Mackeben, M. (1989). Sustained and transient components of focal visual attention. Vision Research, 29(11), 1631-1647.

Nakayama, K., & Silverman, G. H. (1986). Serial and parallel processing of visual feature conjunctions. Nature, 320, 264-265.

Neisser, U. (1967). Cognitive Psychology. New York: Appleton, Century, Crofts.

Nodine, C. F., Krupinski, E. A., & Kundel, H. L. (1993). Visual processing and decision making in search and recognition of targets. In D. Brogan, A. Gale, & K. Carr (Eds.), Visual Search 2, (pp. 239-249). London, UK: Taylor & Francis.

Nothdurft, H. C. (1985). Orientation sensitivity and texture segmentation with different line orientations. Vision Research, 25, 551-560.

Nothdurft, H. C. (1993). Faces and facial expression do not pop-out. Perception, 22, 1287-1298.

Nunnally, J.C. (1978). Psychometric Theory. New York: McGraw-Hill.

O'Neill, P., & Wolfe, J. M. (1997). Individual Differences in Visual Search or Why testing two authors and one naive subject is inadequate. submitted

Ottes, F. P., Van Gisbergen, J. A. M., & Eggermont, J. J. (1986). Visuomotor fields of the superior colliculus: a quantitative model. Vision Research, 26(6), 857-873.

Posner, M. I., & Cohen, Y. (1984). Components of attention. In H. Bouma & D. G. Bouwhuis (Eds.), Attention and Performance X, (pp. 55-66). Hillside, NJ: Erlbaum.

Quinlan, P. T., & Humphreys, G. W. (1987). Visual search for targets defined by combinations of color, shape, and size: An examination of the task constraints on feature and conjunction searches. Perception and Psychophysics, 41, 455- 472.

Reinitz, M. T., Morrissey, J., & Demb, J. (1994). Role of attention in face encoding. Journal of Experimental Psychology: Learning, Memory, and Cognition, 20(1), 161-168.

Rensink, R., O'Regan, J. K., & Clark, J. J. (1995). Visual perception of scene changes disrupted by global transients. Nature, ms.

Robinson, D. (1972). Eye movement evoked by collicular stimulation in the alert monkey. Vision Research, 12, 1795-1808.

Schor, C., & Erickson, P. (1988). Patterns of binocular suppression and accommodation in monovision. Am. J. Optom. and Physiol. Optics, 65, 853- 861.

Schwartz, E. L. (1977). Spatial mapping in primate sensory projection: analytic structure and relevance to perception. Biological Cybernetics, 25, 181-194.

Schwartz, E. L. (1980). Computational anatomy and functional architecture of striate cortex: a spatial mapping approach to perceptual coding. Vision Research, 20, 645-669.

Schwartz, E. L., Desimone, R., Albright, T. D., & Gross, C. G. (1983). Shape recognition and inferior temporal neurons. Proceedings of the National Academy of Science, 80, {5776-5778.

Sheinberg, D. K., & Zelinsky, G. J. (1993). A cortico-collicular model of saccadic target selection. In G. d`Ydewalle & J. Van Rensbergen (Eds.), Perception and Cognition, (pp. 333-348). Amsterdam: Elsevier.

Sparks, D., & Mays, L. (1980). Movement fields of saccade-related burst neurons in the monkey superior colliculus. Brain Research, 190, 39-50.

Spillman, L., & Werner, J. S. (1990). Visual Perception: The Neurophysiological Foundations. San Diego: Academic Press.

Suzuki, S., & Cavanagh, P. (1995). Facial organization blocks access to low-level features: An object inferiority effect. Journal of Experimental Psychology: Human Perception and Performance, 21(4), 901-913.

Swensson, R. G., & Judy, P. F. (1981). Detection of noisy visual targets: Models for the effects of spatial uncertainty and signal -to-noise ratio. Perception and Psychophysics, 29(6), 521-534.

Theeuwes, J. (1995). Parallel search for a conjunction of color and orientation: The effect of spatial proximity. JEP:HPP, ms 95-42 DO NOT CITE.

Theeuwes, J., & Kooi, J. L. (1994). Parallel search for a conjunction of shape and contrast polarity. Vision Research, 34(22), 3013-3016.

Tipper, S. P., Weaver, B., Jerreat, L. M., & Burak, A. L. (1994). Object-based and environment-based inhibition of return of visual attention. Journal of Experimental Psychology: Human Perception & Performance, 20(3), 478-499.

Treisman, A., & Gelade, G. (1980). A feature-integration theory of attention. Cognitive Psychology, 12, 97-136.

Treisman, A., & Sato, S. (1990). Conjunction search revisited. J. Exp. Psychol: Human Perception and Performance, 16(3), 459-478.

Tsotsos, J. K. (1990). Analyzing vision at the complexity level. Brain and Behavioral Sciences, 13(3), 423-469.

Vecera, S. P., & Farah, M. J. (1994). Does visual attention select objects or locations? J. Experimental Psychology: General, 123(2), 146-160.

Visual search for the odd item in a subset. J. Experimental Psychology: Human Perception and Performance, 21(3), 531-551.

Ward, R., Duncan, J., & Shapiro, K. (1996). The slow time-course of visual attention. Cognitive Psychology, in press.

Wolfe, J. M. (1993). Guided Search 2.0: The upgrade of a model of visual search. Investigative Ophthalmology and Visual Science, 34(4), 1289.

Wolfe, J. M. (1994a). Guided Search 2.0: A revised model of visual search. Psychonomic Bulletin and Review, 1(2), 202-238.

Wolfe, J. M. (1994b). The pertinence of research on visual search to radiologic practice. Academic Radiology, 2, 74-78.

Wolfe, J. M. (1994c). Visual search in continuous, naturalistic stimuli. Vision Research, 34(9), 1187-1195.

Wolfe, J. M. (1996). Extending Guided Search: Why Guided Search needs a preattentive "item map". In A. Kramer, G. H. Cole, & G. D. Logan (Eds.), Converging operations in the study of visual selective attention., (pp. 247-270). Washington, DC: American Psychological Association.

Wolfe, J. M. (1997a). Post-attentive vision. Investigative Opthalmology and Visual Science., 37(3), S214.

Wolfe, J. M. (1997b). Visual search. In H. Pashler (Ed.), Attention, . London, UK: University College London Press.

Wolfe, J. M. (1997c). What do 1,000,000 trials tell us about visual search. submitted.

Wolfe, J. M., & Bennett, S. C. (1997). Preattentive Object Files: Shapeless bundles of basic features. Vision Research, 37(1), 25-44.

Wolfe, J. M., & Gancarz, G. (1996). Guided Search 3.0: A model of visual search catches up with Jay Enoch 40 years later. In V. Lakshminarayanan (Ed.), Basic and Clinical Applications of Vision Science, . Dordrecht, Netherlands: Kluwer Academic.

Wolfe, J. M., & Pokorny, C. W. (1990). Inhibitory tagging in visual search: A failure to replicate. Perception and Psychophysics, 48, 357-362.

Wolfe, J. M., Cave, K. R., & Franzel, S. L. (1989). Guided Search: An alternative to the Feature Integration model for visual search. J. Exp. Psychol. - Human Perception and Perf., 15, 419-433.

Wolfe, J. M., Friedman-Hill, S. R., & Bilsky, A. B. (1994). Parallel processing of part/whole information in visual search tasks. Perception and Psychophysics, 55(5), 537-550.

Wolfe, J. M., Friedman-Hill, S. R., Stewart, M. I., & O'Connell, K. M. (1992). The role of categorization in visual search for orientation. J. Exp. Psychol: Human Perception and Performance, 18(1), 34-49.

Wolfe, J. M., O'Neill, P. E., & Bennett, S. C. (1997). Why are there eccentricity effects in visual search? Perception and Psychophysics, in press.

Wolfe, J. M., Yu, K. P., Stewart, M. I., Shorter, A. D., Friedman-Hill, S. R., & Cave, K. R. (1990). Limitations on the parallel guidance of visual search: Color X color and orientation X orientation conjunctions. J. Exp. Psychol: Human Perception and Performance, 16(4), 879-892.

Yantis, S. (1992). Multielement visual tracking: Attention and perceptual organization. Cognitive Psychology, 24, 295-340.

Yantis, S. (1993). Stimulus-driven attentional capture. Current Directions in Psychological Science, 2(5), 156-161.

Yantis, S., & Gibson, B. S. (1994). Object continuity in apparent motion and attention. Canadian Journal of Experimental Psychology, 48, 182-204.

Zelinsky, G. J., & Sheinberg, D. L. (1997). Eye movements during parallel / serial visual search. J. Experimental Psychology: Human Perception and Performance, 23(1), 244-262.

Guided Search 3.0

Jeremy M. Wolfe and Gregory Gancarz

Guided Search 3.0 (GS3) is the third generation of a computational model of visual search behavior. This first part of this document describes the C code used to numerically simulate the GS3 model. The second part is the actual code.

Usage

The program runs on workstations running X Windows. Running the program with no command line flags produces:

```
Usage: s #trials datafile search_type saccade?
        o=orientation feature
        c=color feature
        j=conjunction
        f=findlay (2 target)
        1=conjunction set size 1
        s=serial
        l=large set size conj
```

To run the simulation on 100 trials of a conjunction search with saccadic eye movements, type "s 100 conj.dat j 1".

Program Flow Summary

Do the following for every trial.

Create a search stimulus.

Calculate the Feature Map values.

Do the following until simulation makes a target present or absent response.

Update the salience map.

Is the maximum activation in the salience map > threshold

Yes: Attend to the location of maximum activation

Does attended location contain target features?

Yes: respond Target Present.

No: Inhibit this item.

Are there more salient items?

Yes: continue.

No: respond Target Absent.

No: continue.

Is it time to make a saccade?

Yes: calculate the Saccade Map D layer.

Find the site of maximum activation in the Saccade Map D layer.

Saccade by the amount represented by this site.

Calculate the Feature Map values.

No: continue.

Description of Procedures

set_seed_for_random_generator: by using system time and date, sets a seed for the random number generator.

rnd: returns a number between bottom and top.

msleep: pauses processing.

rk4: numerically integrates differential equations.

save_num_to_file: saves data to a file.

save_2num_to_file: saves data to a file.

concat2: concatenates two strings together.

set_all_colorscells_to_colors: sets colorcells used in drawings to appropriate colors.

draw_level: draws the activation in a layer of the model as a grid where brightness reflects the activation of each unit.

dli: draw a line in the input layer; used for drawing stimuli in the first layer of the model.

draw_rect_in_input: draw a rectangular search item in the input layer.

draw_circle_in_input: draw a circle in the input layer.

draw_early_system: draw a number of model layers to the screen.

draw_cross_on_stim_copy: a copy layer of the stimulus is maintained so that the scanpath of the model can be displayed. This routine draws a cross on that saccade history map when the simulation is running a Findlay type stimulus.

draw_line_on_stim_copy: during a usual visual search task, a number of saccadic eye movements are made. This routine draws a line connecting the eye fixations, thus creating a scanpath on the history map.

decide_next_latency: models saccadic latency as a stochastic process; chooses the latency of the next saccade.

copy_stim_map: copies the activation in the stimulus map to the stimulus history map used for drawing scanpaths and fixation points.

calc_grid_stimulus: draws a grid stimulus to illustrate log scaling used in the model.

calc_ring_stimulus: draws a ring stimulus to illustrate log scaling used in the model.

calc_stimulus: draws a visual search stimulus using oriented lines.

calc_findlay_stimulus: draws a search stimulus similar to that used by Findlay.

init_level: initialize the levels of the model.

clear_level: set the activation of units to 0 in a particular layer.

calc_cone_spacing: calculates the transformation from a Cartesian map to a log mapping used to model the Superior Colliculus (SC).

calc_transformed_map: transforms a Cartesian map to a log map.

calc_cone_act: calculate the activation of the feature maps.

calc_on_c: calculate the response of on-center units.

calc_off_c: calculate the response of off-center units.

calc_vertical_act: calculate the response of vertical detectors.

calc_horizontal_act: calculate the response of horizontal detectors.

getline: used in creating the oriented kernels used by vertical and horizontal detectors.

rect: rectify a number.

calc_vertical_pre_act: calculate vertical activation in a Cartesian map for computational ease; it is then transformed

to a log map by calc_vertical_act.

calc_horizontal_pre_act: calculate horizontal activation in a Cartesian map.

calc_bright_act: calculate bright detector activation.

calc_dark_act: calculate dark detector activation.

calc_sc_superficial: calculate the activation of units representing the Superficial layers of the SC by summing input from the bright and dark detectors.

calc_sc_deep: determine the activation of units representing the deeper SC layers based on the input from the Winner-take-all (WTA) map.

calc_saccadic_target: determine the target of the next saccade by finding the maximally excited SC Deep unit.

calc_color_contrast: calculate bottom-up color activation based on local comparisons between items.

calc_orientation_contrast: calculate bottom-up orientation activation based on local comparisons between items.

calc_weighted_feature_sum_retinal_act: based on top-down settings (what type of features describe the target item), sums the various contributions from the feature maps. These are in a log map.

calc_weighted_feature_sum_world_act: transforms the feature sum log map to world coordinates by taking into consideration eye position.

calc_ior_at: update the activations in the inhibition of return (IOR) map.

feedbk_fn: the signal function between units in the WTA map.

calc_excit: the WTA map had a lower resolution than other maps in the model to save computation. This function calculated the excitatory input to a single WTA unit by adding the activations from units in the feature sum map.

calc_input_to_salience_attn: the input to the WTA map was normalized, and if an input was very small, the input from that unit was set to zero, to save computation.

gate_attn_wta_input_with_ior: the IOR map was multiplied by the WTA input.

sum_attn: add together all the activation in the WTA map.

salience_attention_diffeq: this function contains the differential equation representing a unit in an on-center, off-surround shunting network.

calc_salience_attention_act: update the WTA units and if a unit has sufficient activation, update the attentional window.

calc_features_through_gate: calculate the feature values at the attended location.

calc_do_features_match_target: determine if the features at the attended location match the target features. This function performs simple pattern recognition.

calc_max_map_activity: return the value of maximum activity in a layer of the model.

calc_should_we_saccade: determine if a saccade should be made.

calc_search_result_and_save_data: determine whether model made a correct response, and save data such as reaction time.

calc_fixated_item_type: by comparing the fixated location with the stored position of items, determine which item the simulation is foveating.

adjust_attentional_absent_threshold: adjust the activation threshold using a staircase scheme.

run_early_system: run the routines which only need to be run once per saccade.

set_weights: depending on what the target is, set the top down feature weights appropriately.

run: this procedure calls the procedures responsible for simulating the model. It also takes care of calling the drawing routines.

quit: when the simulation has run the required number of trials, close the X display connection and quit.

get_colors: set up the X colors used in drawing.

get_GC: set up the graphic context used in X drawing.

set_up_x: make a connection with the X server and prepare a window for drawing.

main: decode command line flags (e.g. number of trials to run), and run routines which only need to be run once (e.g. set seed for random number generator). Then enter a loop which calls run, and checks for a keypress. If a keypress is detected, the simulation is paused.

```
/*
 *    Guided Search 3.0 Simulation
 *
 *    Jeremy M. Wolfe and
 *    Gregory Gancarz
 *    All Rights Reserved
 *    Copyright 1996
 *    Supported by:
 *    AFOSR 49620-93-1-0407
 *
 */


/*----------------Include Libraries----------*/

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xos.h>
#include <X11/Xatom.h>
#include <stdio.h>
#include <sys/times.h>
#include <sys/signal.h>
#include <limits.h>
#include <math.h>


/*-----------------Constants-----------------*/

/*output*/
#define         GRAPHICS    1       /*graphics*/
#define         TO          1       /*text output*/

/*colors*/
#define MAX_COLORS 40
#define COLOR_MAX       65535 /*actual max is 65535*/

/*other sim constants*/
#define         PPD                 10      /*map1 pixels per degree of visual angle*/
#define         PPMM                15      /*pixels per mm of cortex*/

/*map sizes*/
#define         NROW1               15*PPD      /*a 15x15 degree field of view*/
#define         NCOLUMN1            15*PPD
#define         NROW2               5*PPMM      /*a 5mm by 5mm piece of brain*/
#define         NCOLUMN2            5*PPMM
#define         MC3                 3           /*used in salience attentional*/

#define         STIMULUS                0
#define         HORIZ_PRE               1
#define         VERT_PRE                2
#define         ON_C                    3
#define         OFF_C                   4
#define         STIM_COPY               5
```

```
#define     CONE                        0
#define     VERTICAL                    1
#define     HORIZONTAL                  2
#define     BRIGHT                      3
#define     DARK                        4
#define     WEIGHTED_FEATURE_SUM_RETINAL 5
#define     IOR_ATTN                    7
#define     SALIENCE_ATTENTIONAL        8
#define     WEIGHTED_FEATURE_SUM_WORLD  9
#define     COLOR_CONTRAST              10
#define ORIENT_CONTRAST                 11
#define     SACCADE_LOCATION            12
#define     ATTN_WTA_INPUT              13
#define     SC_DEEP                     14
#define     SC_SUPER                    6
#define     WTA_RET                     15

#define     NUM_LEVELS                  16

#define     MAX_SET_SIZE                16

/*bottom-up weights*/
double      W_BU_ORIENTATION;
            /*these are set depending on search type is set_weights routine*/
double      W_BU_COLOR;

/*top-down weights indicate what features target has*/
double      W_VERTICAL;
            /*these are set depending on search type is set_weights routine*/
double      W_HORIZONTAL;
double      W_BRIGHT;
double      W_DARK;

/*kernel sizes used to calc orientation maps*/
#define     ORIENT_KERNEL_W             .6      /*width*/
#define     ORIENT_KERNEL_L             1.6     /*length*/

/*used for adjusting attention absent threshold, staircased*/
/*a 7:1 inc:dec ratio seemed to work for past gs simulations*/
#define ABSENT_THRESHOLD_ATTN_INCREMENT    .00003
#define     ABSENT_THRESHOLD_ATTN_DECREMENT    .00021
#define     ABSENT_THRESHOLD_ATTN_MINIMUM      .00400
double      absent_threshold_attn=             .00800;
                    /*if max saliency is <value, respond absent*/

/*range of bottom-up interaction, in mm of cortex*/
#define     BU_DIAM     2

/*drawing window constants*/
#define WIN_WIDTH 150
#define WIN_HEIGHT 150
#define X_BORDER 10
#define Y_BORDER 13
```

```
/*4th order Runge-Kutta step size*/
#define         H       .05             /*step size for rk4*/

/*constants for salience_attention_diffeq*/
#define         AA      .1
#define         BB      3

/*------------------Variables and other Stuff--------------------*/

Display                 *display;
unsigned long colorcell[MAX_COLORS];
GC              *ggc;
GC              dggc;   /*for drawing*/
Display         *gdisplay;
Window          gwin;
int             gscreen_num;
XColor          color;
Colormap        default_cmap;
XColor          set_color[MAX_COLORS];
GC gc;

/*map variables*/
double actl[7][NROW1][NCOLUMN1];
double act[20][NROW2][NCOLUMN2];

int retina_r,retina_c;  /*center of retina location*/
int retina_rh,retina_ch;/*history of fixation (previous location)*/

struct spacings
{
        int r;
        int c;
        int s;          /*scale (radius of square convolution kernel)*/
        int o;          /*does the location have a mapping in retinal space (ok)*/
} space[NROW2][NCOLUMN2];

int attn_r,attn_c;      /*where attention is, in retinotopic coordinates*/
int target_trial;

char search_type;       /*f for feature search, c for conjunction*/
int num_trials=10;      /*number of trials to run*/
int set_size;           /*number of items on the screen*/

int new_trial=1;        /*flag to signify when a new trial is beginning*/
int need_saccade=1;     /*flag to signify when we saccade, so we can calculate early
                                activations only when they are needed (since they are
                                computationally expensive*/

struct items            /*holds the locations for the items which compose the stimulus*/
{                       /*in addition to their type (bright vertical, bright horizontal, or*/
        int r;          /*dark vertical)*/
        int c;
        int type;
} item[MAX_SET_SIZE];
```

```
int present_signal=0,absent_signal=0;

int reaction_time;      /*time simulation takes to respond in ms (not ``real" cpu time)*/
double saccade_latency;
int time_of_last_saccade;/*time when we last did a saccade*/
int time_of_last_gate;  /*time when the gate was last opened*/
char result;            /*C orrect, F alse alarm, or M iss*/

double surround;        /*used in attentional wta map*/

double move_attn_threshold=2.4;
int sim_time;   /*simulation time*/
double vert_act,horiz_act,bright_act,dark_act;

int make_saccades;      /*1 for sim to make saccades, 0 means maintain fixation*/
int trial_num=0;        /*trial counter*/

char filename[100],dwellfile[100],finfile[100],rtfile[100]; /*filenames to use when writing
data*/
int it[8];          /*findlay stim locations*/


/*-------------------------Utility Routines-------------------------*/

void set_seed_for_random_generator()
/*By using system time and date, sets
  seed for random number generator*/
{
  double seed,time,date;
  FILE *fp;
  system("date '+%S%M%H' > seed");
  system("date '+%d%m%y' >> seed");
  fp=fopen("seed","r");
  fscanf(fp,"%f",&time);
  fscanf(fp,"%f",&date);
  seed=time/date;
  fclose (fp);
  system("rm seed");
  srand48(seed);/*was srandom*/
}

double rnd(double bottom, double top)
/*returns a number between bottom and top*/
{
        return ((lrand48()/(double)2147483647)*(top-bottom)+bottom);
}

void msleep(int sec,int usec)
{
        int oldmask,mask;
        struct timeval tv;

        tv.tv_sec=sec;
```

```
        tv.tv_usec=usec;
        select(0,0,0,0,&tv);
}


double rk4(func,x,r,c)
        double (*func)(),x;
        int r,c;
{
    double delta1, delta2, delta3, delta4;

    delta1 = H * (*func)(x,r,c);
    delta2 = H * (*func)(x + delta1/2.0,r,c);
    delta3 = H * (*func)(x + delta2/2.0,r,c);
    delta4 = H * (*func)(x + delta3,r,c);

    return(x+(delta1+2.0*delta2+2.0*delta3+delta4)/6.0);
}

void save_num_to_file(char name[100], double num)
{
        FILE *fp;
        fp=fopen(&name[0],"a");
        fprintf(fp,"%f\n",num);
        fclose(fp);
}

void save_2num_to_file(char name[100], double num1, double num2)
{
        FILE *fp;
        fp=fopen(&name[0],"a");
        fprintf(fp,"%f %f\n",num1,num2);
        fclose(fp);
}

void concat2(char *a, char *b, char *where)
{
    while (*where++=*a++);
    where--;
    while (*where++=*b++);
}


/*-----------------------------Drawing Stuff-----------------------------*/

void set_all_colorscells_to_colors()
{
        int i;
        int level=0;
        for (i=0;i<MAX_COLORS;i++)      /*we set 4th color (white) manually since
used for fixation cross*/
        {
                set_color[i].pixel = colorcell[i];
                set_color[i].flags=7;

    }
```

```
                set_color[i].pad=0;

                set_color[i].red = level;
                set_color[i].green = level;
                set_color[i].blue = level;

                level+=COLOR_MAX/MAX_COLORS;
        }
        XStoreColors(gdisplay, DefaultColormap(gdisplay, gscreen_num), set_color,
MAX_COLORS);
        XFlush(gdisplay);
}

void draw_level(int size, int wlevel, int wr, int wc, int nrow, int ncolumn, char t[100])
{
        int r,c,level;
        double pts;
        double xscale,yscale,max=-99999;
        int x,y,width,height;
        int CX=270;

if (GRAPHICS)
{
        x=wc*(WIN_WIDTH+X_BORDER);
        y=wr*(WIN_HEIGHT+Y_BORDER);

        width=WIN_WIDTH;
        height=WIN_HEIGHT;

        xscale=width/(double)ncolumn;
        yscale=height/(double)nrow;

        /*find max value for normalization*/
        if (size==1)
        {
                for (r=0;r<nrow;r++)
                for (c=0;c<ncolumn;c++)
                        if (actl[wlevel][r][c]>max)
                                max=actl[wlevel][r][c];
                /*printf("wlevel=%d max=%f\n",wlevel,max);*/
        }
        else
        {
                for (r=0;r<nrow;r++)
                for (c=0;c<ncolumn;c++)
                {
                        /*printf("act[%d][%d][%d]=%f  ",wlevel,r,c,max);*/
                        if (act[wlevel][r][c]>max)
                                max=act[wlevel][r][c];
                }
        }

        /*this way, if a color feature search, don't got medium grey scaled to 1 on absent
searches*/
```

```
        if ((search_type=='c' || search_type=='1') && (wlevel==STIMULUS ||
wlevel==CONE))
                        max=1;

        for (r=0;r<nrow;r++)
        for (c=0;c<ncolumn;c++)
        {
                if (size==1)
                        level=(int)((actl[wlevel][r][c]/max)*(MAX_COLORS-1));
                else
                        level=(int)(act[wlevel][r][c]/max*(MAX_COLORS-1));

                XSetForeground(gdisplay, *ggc, colorcell[level]);

                if ((xscale<1) || (yscale<1))
                        XDrawPoint(gdisplay, gwin, dggc,
x+(int)(c*xscale),y+(int)(r*yscale));
                else
                        XFillRectangle(gdisplay, gwin, dggc,
                                x+(int)(c*xscale), y+(int)(r*yscale),
                                (int)xscale+1, (int)(yscale)+1);
        }

        /*draw map names on the screen*/
        XSetForeground(gdisplay, *ggc, colorcell[MAX_COLORS-1]);
        XDrawString(gdisplay,gwin,dggc,x,y+WIN_HEIGHT+10,&t[0],strlen(&t[0]));

        XFlush(gdisplay);
}
}


void dli(r,c,dr,dc,length,val)
/*usage: starting r,c , change r per step, change c per step
  ,#steps */
int r,c,dr,dc,length;
double val;
{
 int i;
 for (i=0;i<length;i++)
 {
        r+=dr;
        c+=dc;
   if ((r>0 && r<(NROW1-1)) && (c>0 && c<(NCOLUMN1-1)))
        {
                actl[STIMULUS][r][c]=val;
        }
 }
}


void draw_rect_in_input(int r,int c,int h,int w,double val)
{
```

```
        int rr;
        for (rr=r;rr<=r+h;rr++)
                dli(rr,c,0,1,w,val);
}


void draw_circle_in_input(int mr,int mc,int rad,double val)
{
        int r,c;
        double dist;
        for (r=mr-rad;r<mr+rad;r++)
        for (c=mc-rad;c<mc+rad;c++)
        {
                dist=sqrt(pow(r-mr,2)+pow(c-mc,2));
                if ((int)dist<rad)
                        actl[STIMULUS][r][c]=val;
        }
}


void draw_early_system()
{
        draw_level(2,CONE,3,2,NROW2,NCOLUMN2,"V1");
        draw_level(2,VERTICAL,2,2,NROW2,NCOLUMN2,"Vertical");
        draw_level(2,HORIZONTAL,2,3,NROW2,NCOLUMN2,"Horizontal");
        draw_level(2,BRIGHT,2,0,NROW2,NCOLUMN2,"Bright");
        draw_level(2,DARK,2,1,NROW2,NCOLUMN2,"Dark");
        draw_level(2,COLOR_CONTRAST,1,1,NROW2,NCOLUMN2,"Bottom-up: C");
        draw_level(2,ORIENT_CONTRAST,1,2,NROW2,NCOLUMN2,"Bottom-up:
O");
        draw_level(2,WEIGHTED_FEATURE_SUM_RETINAL,0,2,NROW2,NCOLUM
N2,"Feature Sum (r)");
        draw_level(2,WEIGHTED_FEATURE_SUM_WORLD,0,5,NROW2,NCOLUM
N2,"Feature Sum (w)");
}


void draw_cross_on_stim_copy()
{
        double n=.2;
        double sac_deg,shift,dist,nr,nc;

        dist=sqrt((retina_r-NROW1/2)*(retina_r-NROW1/2)+
                                (retina_c-NCOLUMN1/2)*(retina_c-NCOLUMN1/2));
        sac_deg=acos((double)(retina_c-NCOLUMN1/2)/dist)*57.3;
        if ((retina_r-NROW1/2)<0)
                sac_deg=360-sac_deg;
        shift=45*(4-it[0]);
        sac_deg=shift+sac_deg;
        nr=NROW1/2+sin(sac_deg/57.3)*dist;
        nc=NCOLUMN1/2+cos(sac_deg/57.3)*dist;

        actl[STIM_COPY][(int)nr][(int)nc]=n;
}
```

```
void draw_line_on_stim_copy()
{
        double dr,dc;
        double r,c;
        int n;
        double dots=100;

        r=retina_rh;
        c=retina_ch;

        dr=(retina_r-retina_rh)/dots;
        dc=(retina_c-retina_ch)/dots;

        for (n=0;n<dots;n++)
        {
                actl[STIM_COPY][(int)r][(int)c]=0;
                r+=dr;
                c+=dc;
        }

        retina_rh=retina_r;
        retina_ch=retina_c;
}

/*----------------------------Calculating Routings----------------------*/

double decide_next_latency()
{
        int n;
        n=rnd(0,6);
        return (200+n*10);
}


void copy_stim_map()
{
        int r,c;
        for (r=0;r<NROW1;r++)
        for (c=0;c<NCOLUMN1;c++)
        {
                actl[STIM_COPY][r][c]=actl[STIMULUS][r][c];
        }
}


void calc_grid_stimulus()
{
        int r,c,dummy;
        int th;
        int line_thickness=10;
        int spacing=50;
```

```
        for (th=0;th<line_thickness;th++)
        {
                for (r=th;r<NROW1;r+=spacing)
                for (c=0;c<NCOLUMN1;c++)
                        actl[STIMULUS][r][c]=1;
                for (c=th;c<NCOLUMN1;c+=spacing)
                for (r=0;r<NROW1;r++)
                        actl[STIMULUS][r][c]=1;
        }
}

void calc_ring_stimulus()
{
        float ra;
        int mr=NROW1/2;
        int mc=NCOLUMN1/2;
        int th;
        float r,c;
        int l;

        for (th=0;th<=360;th++)
        {
                for (ra=0;ra<=90;ra+=15)
                {

        actl[STIMULUS][(int)(cos(th/57.3)*ra)+mr][(int)(sin(th/57.3)*ra)+mc]=1;
                }

                if (th%60==0)
                {
                        r=mr;
                        c=mc;
                        for (l=0;l<90;l++)
                        {
                                actl[STIMULUS][(int)(r)][(int)(c)]=1;
                                r+=cos(th/57.3);
                                c+=sin(th/57.3);
                        }
                }
        }
}


void calc_stimulus()
{
        int r,c,i,j;
        int bar_l=2*PPD,bar_w=.3*PPD;     /*2/3' long by 1/4' wide*/ /*now, 1' long to
make easier at
                                                        lower simulation resolutions (to
increase speed)*/
        double val;
        int trying;
        int X_SPACING=(int)(bar_l*1.8),Y_SPACING=(int)(bar_l*1.8);
        int count;
```

```
double dist_from_center;

int rr,cc;
int mr=(int)(NROW1/X_SPACING);
int mc=(int)(NCOLUMN1/Y_SPACING);

int taken[mr][mc];
int jitter=.9*PPD;

/*printf("bar_l=%d\n",bar_l);*/
/*choose set size*/
i=(int)rnd(0,3);
switch(i)
{
        case 0:
                set_size=5;
                break;
        case 1:
                set_size=10;
                break;
        case 2:
                set_size=15;
                break;
        default:
                printf("ERROR in calc_stimulus\n");
                exit(0);
                break;
}

if (search_type=='1')
        set_size=1;
else if (search_type=='l')
        set_size=15;

/*set bkgd to grey*/
for (r=0;r<NROW1;r++)
for (c=0;c<NCOLUMN1;c++)
        actl[STIMULUS][r][c]=.5;

/*clear taken matrix*/
for (rr=0;rr<mr;rr++)
for (cc=0;cc<mc;cc++)
        taken[rr][cc]=0;

/*choose locations for items*/
for (i=0;i<set_size;i++)
{
        trying=1;
        count=0;
        while (trying)
        {
                trying=0;
                rr=(int)rnd(0,mr);
                cc=(int)rnd(0,mc);
```

```
                item[i].r=rr*X_SPACING + X_SPACING/2 + (int)rnd(-jitter,jitter);
                item[i].c=cc*Y_SPACING + Y_SPACING/2 + (int)rnd(-jitter,jitter);

                /*is this an ok spot for the item (not overlapping other items?)*/
                if (taken[rr][cc])
                {
                        trying=1;
                }
        }
        taken[rr][cc]=1;
}

/*draw items*/
for (i=0;i<set_size;i++)
{

        /*decide what type of item it is*/
        if (target_trial && i==0)
        {
                /*white vertical*/
                item[i].type=0;
                draw_rect_in_input(item[i].r-bar_l/2,item[i].c-bar_w/2,bar_l,bar_w,1);
                /*by subtracting bar_l/2 and bar_w/2, we draw item centered
                        on chosen point*/
        }
        else
        {
                if (search_type=='j' || search_type=='s' || search_type=='l')
                {
                        if (i<(set_size/2))
                        {
                                /*white horizontal*/
                                item[i].type=1;
                                draw_rect_in_input(item[i].r-bar_w/2,item[i].c-
                                        bar_l/2,bar_w,bar_l,1);
                        }
                        else
                        {
                                /*black vertical*/
                                item[i].type=2;
                                draw_rect_in_input(item[i].r-bar_l/2,item[i].c-
                                        bar_w/2,bar_l,bar_w,0);
                        }
                }
                else if (search_type=='o')
                {
                        /*white horizontal*/
                        item[i].type=1;
                        draw_rect_in_input(item[i].r-bar_w/2,item[i].c-
                                bar_l/2,bar_w,bar_l,1);
                }
                else if (search_type=='c')
                {
```

```
                        /*black vertical*/
                        item[i].type=2;
                        draw_rect_in_input(item[i].r-bar_l/2,item[i].c-
                                bar_w/2,bar_l,bar_w,0);
                }
                if (search_type=='1')
                {
                        if (rnd(0,1)>.5)
                        {
                                /*white horizontal*/
                                item[i].type=1;
                                draw_rect_in_input(item[i].r-bar_w/2,item[i].c-
                                        bar_l/2,bar_w,bar_l,1);
                        }
                        else
                        {
                                /*black vertical*/
                                item[i].type=2;
                                draw_rect_in_input(item[i].r-bar_l/2,item[i].c-
                                        bar_w/2,bar_l,bar_w,0);
                        }
                }
        }
    }
}


void calc_findlay_stimulus()
{
        int r,c;
        double radius=3.7*PPD;
        double circ_rad=.5*PPD;      /*.375 is what Findlay used in '95 paper*/
        double theta[8]={0,45,90,135,180,225,270,315};
        int taken[8];
        int mr,mc;
        int s,i;
        int loc=0;
        double rshift=0,cshift=0;

        /*set bkgd to grey*/
        for (r=0;r<NROW1;r++)
        for (c=0;c<NCOLUMN1;c++)
                actl[STIMULUS][r][c]=.5;

        for (i=0;i<8;i++)
        {
                /*theta[i]+=22.5;*/
                taken[i]=0;
        }

        if (trial_num==1)
                it[0]=4;
        else
                it[0]=(int)rnd(1,7);     /*not 0,8 since if 8, then it[0]+1 would be >8...*/
```

```
        it[1]=it[0]+1;
        taken[it[0]]=1;
        taken[it[1]]=1;

        for (i=2;i<8;i++)
        {
                while (taken[loc])
                {
                        loc++;
                }
                it[i]=loc;
                loc++;
        }

        rshift=-1;
        cshift=0;
        for (s=0;s<8;s++)
        {
                /*printf("%f\n",theta[it[s]]);*/
                item[s].r=NROW1/2 + sin(theta[it[s]]/57.3)*radius+rshift;
                item[s].c=NCOLUMN1/2 + cos(theta[it[s]]/57.3)*radius+cshift;

                if (s==0)
                        draw_circle_in_input(item[s].r,item[s].c,circ_rad,1);
                else if (s==1)
                        draw_circle_in_input(item[s].r,item[s].c,circ_rad,1);
                else
                        draw_circle_in_input(item[s].r,item[s].c,circ_rad,0);
        }
}


void init_levels()
{
        int r,c,l;
        for (l=0;l<=NUM_LEVELS;l++)
        {
                for (r=0;r<NROW2;r++)
                for (c=0;c<NCOLUMN2;c++)
                {
                        if (l==IOR_ATTN)
                                act[l][r][c]=1;
                        else
                                act[l][r][c]=0;
                }
        }
}

void clear_level(int which)
{
        int r,c;
        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
```

```
            if (which==IOR_ATTN)
                act[which][r][c]=1;
            else
                act[which][r][c]=0;
        }
}


void calc_cone_spacing()
{
        int r,c;
        int mr,mc;      /*middle r and c*/
        double dr,dc;   /*difference, in mm*/
        double rad,theta;
        double vr,vc;
        double A=3;
        double Bc=1.4;          /*in mm*/
        double Br=1.8;          /*in mm/deg*/
        int ok;

        mr=NROW2/2;
        mc=NCOLUMN2/2;


        for (c=mc;c<NCOLUMN2-1;c++)
        {
                ok=1;
                for (r=mr;r>=0;r--)
                {
                        /*printf("r=%d\n",r);*/

                        if (ok)
                        {
                                dr=r-mr;        /*in pixels*/
                                dc=c-mc;

                                dr=dr/PPMM;  /*now in mm*/
                                dc=dc/PPMM;          /*now in mm*/

                                rad=A*sqrt(exp(2*dc/Bc)-2*exp(dc/Bc)*cos(dr/Br) + 1);
        /*in deg*/
                                theta=atan(exp(dc/Bc)*sin(dr/Br)/(exp(dc/Bc)*cos(dr/Br)-1));
                                        /*in radians*/

                                vr=rad*sin(theta);      /*in deg of visual angle*/
                                vc=rad*cos(theta);      /*in deg*/

                                vr=vr*PPD;
                                vc=vc*PPD;   /*now in pixels*/

                                /*printf("dc=%f (mm), dr=%f (mm), rad=%f(deg), theta=%f (rad)\n",
                                        dc,dr,rad,theta);*/

                                if (theta>0)
```

```
                        {
                                /*printf(">0\n");*/
                                ok=0;
                        }

                        space[r][c].r=vr;
                        space[r][c].c=vc;
                        space[r][c].s=0;

                        /*vert mirror*/
                        space[2*mr-r][c].r=-vr;
                        space[2*mr-r][c].c=vc;
                        space[2*mr-r][c].s=0;

                        /*horiz mirror*/
                        space[r][2*mc-c].r=vr;
                        space[r][2*mc-c].c=-vc;
                        space[r][2*mc-c].s=0;

                        /*diag mirror*/
                        space[2*mr-r][2*mc-c].r=-vr;
                        space[2*mr-r][2*mc-c].c=-vc;
                        space[2*mr-r][2*mc-c].s=0;
                }
                space[r][c].o=ok;
                space[2*mr-r][c].o=ok;
                space[r][2*mc-c].o=ok;
                space[2*mr-r][2*mc-c].o=ok;

        }
    }
}


void calc_transformed_map(int source_map, int target_map)
{
    int r,c;
    int sr,sc;      /*scaled r and c*/

    for (r=0;r<NROW2;r++)
    for (c=0;c<NCOLUMN2;c++)
    {
        if (space[r][c].o)
        {
            sr=retina_r+space[r][c].r;
            sc=retina_c+space[r][c].c;

            if ((sr<0) || (sr>=NROW1) || (sc<0) || (sc>=NCOLUMN1))
                act[target_map][r][c]=0;
            else
            {
                act[target_map][r][c]=actl[source_map][sr][sc];
            }
        }
```

```
        }
}


void calc_cone_act()
{
        int r,c;
        int sr,sc;      /*scaled r and c*/
        int border=0;

        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                if (space[r][c].o)
                {
                        sr=retina_r+space[r][c].r;
                        sc=retina_c+space[r][c].c;

                        if ((sr<0+border) || (sr>=NROW1-border) || (sc<0+border) ||
(sc>=NCOLUMN1-border))
                                act[CONE][r][c]=.5;
                        else
                        {
                                act[CONE][r][c]=actl[STIMULUS][sr][sc];
                        }
                }
        }
}


void calc_on_c()
{
        int r,c;
        int ir,ic;
        int kr,kc;
        int nkr=3,nkc=3;
        double k[3][3]={-1,-1,-1,-1,14,-1,-1,-1,-1};
        double tot;
        int pr,pc;

        for (r=0;r<NROW1;r++)
        for (c=0;c<NCOLUMN1;c++)
        {
                tot=0;

                ir=r-nkr/2;
                ic=c-nkc/2;

                /*run through kernel*/
                for (kr=0;kr<nkr;kr++)
                for (kc=0;kc<nkc;kc++)
                {
                        pr=ir+kr;
                        pc=ic+kc;
```

```
                if (pr>=0 && pr<NROW1 && pc>=0 && pc<NCOLUMN1)
                {
                        tot+=actl[STIMULUS][pr][pc]*k[kr][kc];
                }
            }
            if (tot<5)
                tot=0;
            actl[ON_C][r][c]=tot/8.5;
        }
    }


void calc_off_c()
{
        int r,c;
        int ir,ic;
        int kr,kc;
        int nkr=3,nkc=3;
        double k[3][3]={1,1,1,1,-14,1,1,1,1};
        double tot;
        int pr,pc;

        for (r=0;r<NROW1;r++)
        for (c=0;c<NCOLUMN1;c++)
        {
            tot=0;

            ir=r-nkr/2;
            ic=c-nkc/2;

            /*run through kernel*/
            for (kr=0;kr<nkr;kr++)
            for (kc=0;kc<nkc;kc++)
            {
                pr=ir+kr;
                pc=ic+kc;

                if (pr>=0 && pr<NROW1 && pc>=0 && pc<NCOLUMN1)
                {
                        tot+=actl[STIMULUS][pr][pc]*k[kr][kc];
                }
            }
            tot+=6;
            if (tot<5)
                tot=0;
            actl[OFF_C][r][c]=tot/8.5;
        }
}

void calc_vertical_act()
{
        int r,c;
        int sr,sc;      /*scaled r and c*/
```

```
        int border=20;

        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                if (space[r][c].o)
                {
                        sr=retina_r+space[r][c].r;
                        sc=retina_c+space[r][c].c;

                        if ((sr<0) || (sr>=NROW1) || (sc<0) || (sc>=NCOLUMN1))
                                act[VERTICAL][r][c]=0;
                        else
                        {
                                act[VERTICAL][r][c]=actl[VERT_PRE][sr][sc];
                        }
                }
        }
}

void calc_horizontal_act()
{
        int r,c;
        int sr,sc;      /*scaled r and c*/
        int border=20;

        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                if (space[r][c].o)
                {
                        sr=retina_r+space[r][c].r;
                        sc=retina_c+space[r][c].c;

                        if ((sr<0) || (sr>=NROW1) || (sc<0) || (sc>=NCOLUMN1))
                                act[HORIZONTAL][r][c]=0;
                        else
                        {
                                act[HORIZONTAL][r][c]=actl[HORIZ_PRE][sr][sc];
                        }
                }
        }
}

double getline(r,c,dr,dc,length)
/*usage: starting r,c , change r per step, change c per step
  ,#steps */
int r,c,dr,dc,length;
{
        int i;
        double sum=0;
        for (i=0;i<length;i++)
        {
                if (r>=0 && r<NROW1 && c>=0 && c<NCOLUMN1 )
```

```
                        sum+=actl[ON_C][r][c]+actl[OFF_C][r][c];
                r+=dr;
                c+=dc;
        }
        return (sum);
}


double rect(double num)
{
        if (num<0)
                num=0;
        return(num);
}


void calc_vertical_pre_act()
{
        /*vertical detectors*/
        int i,r,c,row,column,sh;
        int ht=ORIENT_KERNEL_L*PPD/2;
        int wd=ORIENT_KERNEL_W*PPD/2;
                /*divided by 2 since wd and ht are actually doubled when computing the filter*/
        double s1,s2;

        /*printf("ht=%d wd=%d\n",ht,wd);*/
        if (ht<1)
                ht=1;
        if (wd<1)
                wd=1;
        for (r=0;r<NROW1;r++)
        for (c=wd;c<NCOLUMN1-wd;c++)
        {
                s1=0;
                s2=0;

                /*noon*/
                for (sh=-wd;sh<wd;sh++)
                        s1+=getline(r-ht,c+sh,1,0,ht*2);
                /*if (s1>20)
                        printf("s1=%f ",s1);*/
                if (s1>25)
                        actl[VERT_PRE][r][c]=s1/40;
                else
                        actl[VERT_PRE][r][c]=0;
        }
}

void calc_horizontal_pre_act()
{
        /*horizontal detectors*/
        int i,r,c,row,column,sh;
        int ht=ORIENT_KERNEL_L*PPD/2;
        int wd=ORIENT_KERNEL_W*PPD/2;       /*divided by 2 since wd and ht
```

```
                are actually doubled when computing the filter*/
        double s1,s2;

        if (ht<1)
                ht=1;
        if (wd<1)
                wd=1;
        for (r=wd;r<NROW1-wd;r++)
        for (c=0;c<NCOLUMN1;c++)
        {
                s1=0;
                s2=0;

                for (sh=-wd;sh<wd;sh++)
                        s1+=getline(r+sh,c-ht,0,1,ht*2);
                if (s1>25)
                        actl[HORIZ_PRE][r][c]=s1/40;
                else
                        actl[HORIZ_PRE][r][c]=0;
        }
}


void calc_bright_act()
{
        calc_transformed_map(ON_C,BRIGHT);
}


void calc_dark_act()
{
        calc_transformed_map(OFF_C,DARK);
}

void calc_sc_superficial()
{
        int r,c;
        int kr,kc;
        double dist;
        double diam=3*PPMM;

        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                if ((act[BRIGHT][r][c]>.001) || (act[DARK][r][c]>.001))
                {
                        for (kr=r-diam/2;kr<r+diam/2;kr++)
                        for (kc=c-diam/2;kc<c+diam/2;kc++)
                        {
                                dist=sqrt((kr-r)*(kr-r) + (kc-c)*(kc-c));
                                if (dist<diam/2)
                                if (kr>=0 && kr<NROW2 && kc>=0 &&
kc<NCOLUMN2 && space[kr][kc].o)
                                        act[SC_SUPER][kr][kc]+=
```

```
                                        (act[BRIGHT][r][c]+act[DARK][r][c])*
                                        1/(exp(dist*dist/160));
                        }
                }
        }
        /*add a little noise*/
        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                if (space[r][c].o)
                        act[SC_SUPER][r][c]+=rnd(0,.5);
        }
}


void calc_sc_deep()
{
        int r,c;
        double sr,sc;
        double conv;
        double rscaler,cscaler;
        double nr1=NROW1,nr2=NROW2,nc1=NCOLUMN1,nc2=NCOLUMN2;
        int kr,kc;
        double dist;
        double diam=2*PPMM;

        rscaler=nr2/nr1;
        cscaler=nc2/nc1;

        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                act[SC_DEEP][r][c]=0;/*.1*act[SC_SUPER][r][c];*/
                act[WTA_RET][r][c]=0;       /*clear WTA_RET so += will start fresh*/
        }

        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                if (space[r][c].o)
                {
                        sr=retina_r+space[r][c].r;
                        sc=retina_c+space[r][c].c;

                        sr=sr*rscaler;
                        sc=sc*cscaler;

                        /*printf("sr=%f sc=%f\n",sr,sc);*/

                        if ((sr>=0) && (sr<NROW2) && (sc>=0) && (sc<NCOLUMN2))

        act[WTA_RET][r][c]+=act[SALIENCE_ATTENTIONAL][(int)sr][(int)sc];
                }
        }
```

```
        /*now, do blurring in cortical (sc) coordinates (mm)*/
        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                if (act[WTA_RET][r][c]>.001)
                {
                        for (kr=r-diam/2;kr<r+diam/2;kr++)
                        for (kc=c-diam/2;kc<c+diam/2;kc++)
                        {
                                dist=sqrt((kr-r)*(kr-r) + (kc-c)*(kc-c));
                                if (dist<diam/2)
                                if (kr>=0 && kr<NROW2 && kc>=0 &&
kc<NCOLUMN2 && space[kr][kc].o)
                                        act[SC_DEEP][kr][kc]+=
                                                act[WTA_RET][r][c]*
                                                1/(exp(dist*dist/90));
                        }
                }
        }

        /*add a little noise*/
        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                if (space[r][c].o)
                        act[SC_DEEP][r][c]+=rnd(0,.05);
        }
}


void calc_saccadic_target()
{
        int r,c;
        float max=-999;
        int saccade_target_r,saccade_target_c;

        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                if (act[SC_DEEP][r][c]>max)
                {
                        max=act[SC_DEEP][r][c];
                        saccade_target_r=r;
                        saccade_target_c=c;
                }
        }

        retina_r+=space[saccade_target_r][saccade_target_c].r;
        retina_c+=space[saccade_target_r][saccade_target_c].c;

        if (retina_r<0)
                retina_r=0;
        if (retina_r>NROW1)
```

```
            retina_r=NROW1;
        if (retina_c<0)
            retina_c=0;
        if (retina_c>NCOLUMN1)
            retina_c=NCOLUMN1;
}


void calc_color_contrast()
{
        int r,c,rr,cc;
        double dark,bright;
        int rds=BU_DIAM*PPMM;

        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
            act[COLOR_CONTRAST][r][c]=0;
            if (act[BRIGHT][r][c]>.8 || act[DARK][r][c]>.8)
            {
                dark=0;
                bright=0;
                for (rr=r-rds;rr<r+rds;rr++)
                for (cc=c-rds;cc<c+rds;cc++)
                {
                    if (rr>=0 && rr<NROW2 && cc>=0 && cc<NCOLUMN2)
                    {
                        bright+=act[BRIGHT][rr][cc];
                        dark+=act[DARK][rr][cc];
                    }
                }

        act[COLOR_CONTRAST][r][c]=act[BRIGHT][r][c]*dark+act[DARK][r][c]*bright;
                    /*act[COLOR_CONTRAST][r][c]=rnd(0,1);*/
            }
        }
}


void calc_orientation_contrast()
{
        int r,c,rr,cc;
        double vert,horiz;
        int rds=BU_DIAM*PPMM;
        double dist;
        double min_dist=3*PPMM;
        double max_dist=12*PPMM;

        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
            act[ORIENT_CONTRAST][r][c]=0;
            if (act[VERTICAL][r][c]>.2 || act[HORIZONTAL][r][c]>.2)
            {
```

```
                    vert=0;
                    horiz=0;
                    for (rr=r-rds;rr<r+rds;rr++)
                    for (cc=c-rds;cc<c+rds;cc++)
                    {
                            dist=sqrt((rr-r)*(rr-r) + (cc-c)*(cc-c));
                            if (dist>min_dist && dist<max_dist)
                            if (rr>=0 && rr<NROW2 && cc>=0 && cc<NCOLUMN2)
                            {
                                    vert+=act[VERTICAL][rr][cc];
                                    horiz+=act[HORIZONTAL][rr][cc];
                            }
                    }

        act[ORIENT_CONTRAST][r][c]=act[VERTICAL][r][c]*horiz+act[HORIZONTAL][r][c]*vert;
                }
        }
}


void calc_weighted_feature_sum_retinal_act()
{
        int r,c;

        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                act[WEIGHTED_FEATURE_SUM_RETINAL][r][c]=
                        W_BU_ORIENTATION*act[ORIENT_CONTRAST][r][c]+
                        W_BU_COLOR*act[COLOR_CONTRAST][r][c]+
                        W_VERTICAL*act[VERTICAL][r][c]+
                        W_HORIZONTAL*act[HORIZONTAL][r][c]+
                        W_BRIGHT*act[BRIGHT][r][c]+
                        W_DARK*act[DARK][r][c];
        }
}


void calc_weighted_feature_sum_world_act()
{
        int r,c;
        double rr,rc;   /*retinal r,c*/
        double rscaler,cscaler;
        int kr,kc;
        int diam=10;
        double dist;
        double nr1=NROW1,nr2=NROW2,nc1=NCOLUMN1,nc2=NCOLUMN2;

        rscaler=nr2/nr1;
        cscaler=nc2/nc1;

        /*printf("rscaler=%f cscaler=%f\n",rscaler,cscaler);*/

        for (r=0;r<NROW2;r++)
```

```
        for (c=0;c<NCOLUMN2;c++)
                act[WEIGHTED_FEATURE_SUM_WORLD][r][c]=0;

        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                if (act[WEIGHTED_FEATURE_SUM_RETINAL][r][c]>.00001)
                {
                        rr=retina_r+space[r][c].r;
                        rc=retina_c+space[r][c].c;

                        rr=rr*rscaler;
                        rc=rc*cscaler;

                        /*printf("rr=%f, rc=%f\n",rr,rc);*/
                        /*act[WEIGHTED_FEATURE_SUM_WORLD][(int)rr][(int)rc]=
                                act[WEIGHTED_FEATURE_SUM_RETINAL][r][c]+rnd(0,.01);*/
                        /*in above, if we used +=, then the center fixated item gets very high act,
                                and peripheral items can't be seen.  using just an = reduces the
                                strength of the center item since some points over-write each other
                                I think*/

                        for (kr=rr-diam/2;kr<rr+diam/2;kr++)
                        for (kc=rc-diam/2;kc<rc+diam/2;kc++)
                        {
                                dist=sqrt((kr-rr)*(kr-rr) + (kc-rc)*(kc-rc));
                                if (dist<diam/2)
                                if (kr>=0 && kr<NROW2 && kc>=0 &&
kc<NCOLUMN2)

        act[WEIGHTED_FEATURE_SUM_WORLD][(int)kr][(int)kc]+=

        act[WEIGHTED_FEATURE_SUM_RETINAL][r][c]*
                                        1/(exp(dist*dist/200));
                        }
                }
        }

        /*add a little noise to break symmetry*/
        /*we also add in a little noise so on findly stim, no two points will have exactly
                the same activation (then wta wouldn't work)*/
        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
                act[WEIGHTED_FEATURE_SUM_WORLD][r][c]+=rnd(0,.002);
}

void calc_ior_at(int wr, int wc)
{
        int r,c;
        int diam;
        double dist;
        double val=0;

        diam=1.4*PPMM;
```

```
        /*printf("diam=%d\n",diam);*/

        /*clear the area*/
        for (r=wr-diam/2;r<wr+diam/2;r++)
        for (c=wc-diam/2;c<wc+diam/2;c++)
        {
                dist=sqrt((wr-r)*(wr-r) + (wc-c)*(wc-c));
                if (dist<diam/2)
                        if (r>=0 && r<NROW2 && c>=0 && c<NCOLUMN2)
                        {
                                act[IOR_ATTN][r][c]=0;
                                /*val=1-dist/radius;
                                if (val<0)
                                        val=0;
                                act[IOR_ATTN][r][c]-=val;
                                if (act[IOR_ATTN][r][c]<0)
                                        act[IOR_ATTN][r][c]=0;*/
                        }
        }
}


/*------------attentional salience-----------------*/


double feedbk_fn(double xx)
{
     return (xx*xx);
}


double calc_excit(r,c)
{
        int rr,cc;
        double sum=0;

        for (rr=r;rr<r+MC3;rr++)
        for (cc=c;cc<c+MC3;cc++)
                sum+=act[WEIGHTED_FEATURE_SUM_WORLD][rr][cc];
        return (sum);
}

void calc_input_to_salience_attn()
{
        double tot=0;
        int r,c,rr,cc;
        int diam=8;
        double dist;
        double max=-999;
        double inhib=7000;
        double excit;
        double A=.1, B=1;

        /*printf("inhib=%f\n",inhib);*/
```

```
        for (r=0;r<NROW2;r+=MC3)
        for (c=0;c<NCOLUMN2;c+=MC3)
        {
                if (act[WEIGHTED_FEATURE_SUM_WORLD][r][c]>.0001)
                {
                        excit=calc_excit(r,c);
                        act[ATTN_WTA_INPUT][r][c]=B*excit/(A+excit+inhib);
                }
        }
}


void gate_attn_wta_input_with_ior()
{
        int r,c;
        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {

        act[ATTN_WTA_INPUT][r][c]=act[ATTN_WTA_INPUT][r][c]*act[IOR_ATTN]
[r][c];
        }
}


double sum_attn()
{
        int r,c;
        double sum=0;
        for (r=0;r<NROW2;r+=MC3)
        for (c=0;c<NCOLUMN2;c+=MC3)
        {
                sum+=feedbk_fn(act[SALIENCE_ATTENTIONAL][r][c]);
        }
        return (sum);
}


double salience_attention_diffeq(double x, int r, int c)
{
    return (-AA*x + (BB-x)*(feedbk_fn(x)+act[ATTN_WTA_INPUT][r][c]) -
        x*(surround-feedbk_fn(x)));
}


int calc_salience_attention_act()
{
        int r,c,rr,cc;
        int num;

        for (num=0;num<7;num++)
        {
                surround=sum_attn();
```

```
		for (r=0;r<NROW2;r+=MC3)
		for (c=0;c<NCOLUMN2;c+=MC3)
		if (act[ATTN_WTA_INPUT][r][c]>.001)
		{

	act[SALIENCE_ATTENTIONAL][r][c]=rk4(salience_attention_diffeq,
				act[SALIENCE_ATTENTIONAL][r][c],r,c);
			if (act[SALIENCE_ATTENTIONAL][r][c]>move_attn_threshold)
			{
				if (TO)
					printf(">move_attn_threshold\n");
				attn_r=r;
				attn_c=c;
				return(1);
			}
			for (rr=r;rr<r+MC3;rr++)
			for (cc=c;cc<c+MC3;cc++)

	act[SALIENCE_ATTENTIONAL][rr][cc]=act[SALIENCE_ATTENTIONAL][r][c]
;
		}
	}
	draw_level(2,SALIENCE_ATTENTIONAL,1,5,NROW2,NCOLUMN2,"Salience
(attn)");
	return(0);	/*0 means no item activity above threshold*/
}

/*----------------------*/


void calc_features_through_gate()
{
	int wr,wc,r,c;
	double g;
	double nr1=NROW1,nr2=NROW2,nc1=NCOLUMN1,nc2=NCOLUMN2;
	double rscaler,cscaler;

	vert_act=0;
	horiz_act=0;
	bright_act=0;
	dark_act=0;

	rscaler=nr2/nr1;
	cscaler=nc2/nc1;

	for (r=0;r<NROW2;r++)
	for (c=0;c<NCOLUMN2;c++)
	{
		wr=(retina_r+space[r][c].r)*rscaler;
		wc=(retina_c+space[r][c].c)*cscaler;

		if (wr>=0 && wr<NROW2 && wc>=0 && wc<NCOLUMN2)
		{
			g=act[SALIENCE_ATTENTIONAL][wr][wc];
```

```
                        vert_act+=act[VERTICAL][r][c]*g;
                        horiz_act+=act[HORIZONTAL][r][c]*g;
                        bright_act+=act[BRIGHT][r][c]*g;
                        dark_act+=act[DARK][r][c]*g;
                }
        }

        if (TO)
                printf("v=%f h=%f b=%f d=%f\n",
                        vert_act,horiz_act,bright_act,dark_act);
}


int calc_do_features_match_target()
{
        if (search_type=='f')
        {
                if (bright_act>dark_act)
                {
                        if (TO)
                                printf("Feature Match\n");
                        return(1);
                }
        }
        else if ((vert_act>horiz_act) && (bright_act>dark_act))
        {
                /*if we have strong vert and bright signal, and no
other strong signal, say yes*/
                if (TO)
                        printf("Feature Match\n");
                return(1);
        }
        return(0);      /*0 means that the features didn't match the target's features*/
}


double calc_max_map_activity(int map)
{
        int r,c;
        double max=-999;
        for (r=0;r<NROW2;r++)
        for (c=0;c<NCOLUMN2;c++)
        {
                /*printf("%f\n",act[map][r][c]);*/
                if (act[map][r][c]>max)
                {
                        max=act[map][r][c];
                }
        }
        /*printf("max map activity=%f\n",max);*/
        return(max);
}
```

```
int calc_should_we_saccade()
{
        if ((sim_time-time_of_last_saccade)>saccade_latency)
                return(1);
        else
                return(0);
}


void calc_search_result_and_save_data()
{
        FILE *fp;
        char target;
        double ecc;
        double sac_deg;
        double deg_error,tar0,tar1,err0,err1,dist;

        /*the simulation has made a decision as to target presence*/
        if (present_signal)     /*said present*/
        {
                if (target_trial)
                {
                        result='C';
                        if (TO)
                                printf("Correct\n");
                }
                else
                {
                        result='F';
                        if (TO)
                                printf("False Alarm\n");
                }
        }
        else    /*said absent*/
        {
                if (target_trial)
                {
                        result='M';
                        if (TO)
                                printf("Miss\n");
                }
                else
                {
                        result='C';
                        if (TO)
                                printf("Correct\n");
                }
        }

        /*convert numerical target present var to character for ease of reading data file*/
        if (target_trial)
                target='P';
        else
                target='A';
```

```
        /*figure out target eccentricity*/
        ecc=sqrt(pow(NROW1/2-item[0].r,2)+pow(NCOLUMN1/2-item[0].c,2))/PPD;

        /*save the data to a file*/
        /*for simplicity, we'll update incrementally, even though it's slower*/
        fp=fopen(rtfile,"a");
        fprintf(fp,"%d %c %c %d %f
%f\n",set_size,target,result,reaction_time,absent_threshold_attn,
                ecc);
        fclose(fp);
        if (TO)
                printf("%d %c %c %d %f
%f\n",set_size,target,result,reaction_time,absent_threshold_attn,
                ecc);

        /*if we are doing a "findlay" trial, save some extra data*/
        if (search_type=='f')
        {
                /*calc the deg error*/
                dist=sqrt((retina_r-NROW1/2)*(retina_r-NROW1/2)+
                                        (retina_c-NCOLUMN1/2)*(retina_c-NCOLUMN1/2));
                sac_deg=acos((double)(retina_c-NCOLUMN1/2)/dist)*57.3;
                if ((retina_r-NROW1/2)<0)
                        sac_deg=360-sac_deg;

                tar0=it[0]*45;
                tar1=it[1]*45;

                err0=fabs(tar0-sac_deg);
                err1=fabs(tar1-sac_deg);

                if (err0<err1)
                        deg_error=err0;
                else
                        deg_error=err1;

                fp=fopen(finfile,"a");
                fprintf(fp,"%f %f\n",deg_error,saccade_latency);
                fclose(fp);

                if (TO)
                        printf("sac_deg=%f\n",sac_deg);
        }
}


void calc_fixated_item_type()
{
        int i;
        int dr,dc;
        double dist;    /*Euclidean distance between fixation point and item centers*/
        double min_dist=999999;
        int closest_item=0;
```

```
        for (i=0;i<set_size;i++)
        {
                dr=item[i].r-retina_r;
                dc=item[i].c-retina_c;
                dist=sqrt(dr*dr + dc*dc);
                if (dist<min_dist)
                {
                        min_dist=dist;
                        closest_item=i;
                }
        }
        /*printf("Item type: %d\n",item[closest_item].type);*/
        printf("Fixated Item: ");
        switch (item[closest_item].type)
        {
                case 0:
                        printf("WV\n");
                        break;
                case 1:
                        printf("WH\n");
                        break;
                case 2:
                        printf("BV\n");
                        break;
                default:
                        printf("Error\n");
                        break;
        }
}


void adjust_attentional_absent_threshold()
{
        if (result=='C')
                absent_threshold_attn+=ABSENT_THRESHOLD_ATTN_INCREMENT;
        else
                absent_threshold_attn-=ABSENT_THRESHOLD_ATTN_DECREMENT;
        if (absent_threshold_attn<ABSENT_THRESHOLD_ATTN_MINIMUM)
                absent_threshold_attn=ABSENT_THRESHOLD_ATTN_MINIMUM;
        if (TO)
                printf("absent_threshold_attn=%f\n",absent_threshold_attn);
}


void run_early_system()
{
        calc_cone_act();
        calc_vertical_act();
        calc_horizontal_act();
        calc_bright_act();
        calc_dark_act();
        /*calc_color_contrast();
        calc_orientation_contrast();*/
```

```
        calc_weighted_feature_sum_retinal_act();
        calc_weighted_feature_sum_world_act();

}

void set_weights()
{
        if (search_type=='o')
        {
                W_BU_COLOR=0;
                W_BU_ORIENTATION=0;

                W_VERTICAL=1;
                W_HORIZONTAL=.2;
                W_BRIGHT=0;
                W_DARK=0;
        }
        else if (search_type=='c')
        {
                W_BU_COLOR=0;
                W_BU_ORIENTATION=0;

                W_VERTICAL=.3;
                W_HORIZONTAL=0;
                W_BRIGHT=1.7;
                W_DARK=0;
        }
        else if (search_type=='j')
        {
                W_BU_COLOR=0;
                W_BU_ORIENTATION=0;

                W_VERTICAL=1;
                W_HORIZONTAL=0;
                W_BRIGHT=1;
                W_DARK=0;
                /*absent_threshold_attn=.0100;*/
        }
        else if (search_type=='1')
        {
                W_BU_COLOR=0;
                W_BU_ORIENTATION=0;

                W_VERTICAL=.8;
                W_HORIZONTAL=.2;
                W_BRIGHT=.8;
                W_DARK=.2;
                /*absent_threshold_attn=.0110;*/
        }
        else if (search_type=='f')
        {
                W_BU_COLOR=0;
                W_BU_ORIENTATION=0;
```

```
		W_VERTICAL=0;
		W_HORIZONTAL=0;
		W_BRIGHT=1;
		W_DARK=0;
	}
	else if (search_type=='s')
	{
		W_BU_COLOR=0;
		W_BU_ORIENTATION=0;

		W_VERTICAL=.5;
		W_HORIZONTAL=.5;
		W_BRIGHT=.5;
		W_DARK=.5;
	}
	else if (search_type=='l')
	{
		W_BU_COLOR=0;
		W_BU_ORIENTATION=0;

		W_VERTICAL=1;
		W_HORIZONTAL=0;
		W_BRIGHT=1;
		W_DARK=0;
		/*absent_threshold_attn=.0100;*/
	}
}


int run()
{
	if (new_trial)
	{
		if (TO)
			printf("---------------------------------------\n");
		else
			printf("%d\n",trial_num);

		if (trial_num>=num_trials)
		{
			printf("All done.\n");
			exit(1);
		}
		trial_num++;

		if (GRAPHICS)
			XClearWindow(gdisplay,gwin);

		init_levels();

		/*run sim on a search stimulus*/
		target_trial=(int)rnd(0,2);
		if (search_type=='f')
		{
```

```
                calc_findlay_stimulus();
                if (trial_num==1)
                {
                        /*to keep track of fixation locations in 1 plot*/
                        copy_stim_map();
                }
        }
        else
        {
                calc_stimulus();
                /*to keep track of fixation locations in 1 plot*/
                copy_stim_map();
        }
        /*calc_grid_stimulus();*/
        /*calc_ring_stimulus();*/

        draw_level(1,STIMULUS,4,2,NROW1,NCOLUMN1,"Image");
        draw_level(1,STIM_COPY,4,0,NROW1,NCOLUMN1,"Image");

        calc_on_c();
        calc_off_c();
        draw_level(1,ON_C,3,0,NROW1,NCOLUMN1,"On-C");
        draw_level(1,OFF_C,3,1,NROW1,NCOLUMN1,"Off-C");

        calc_horizontal_pre_act();
        calc_vertical_pre_act();
        draw_level(1,VERT_PRE,3,3,NROW1,NCOLUMN1,"Vert_pre");
        draw_level(1,HORIZ_PRE,3,4,NROW1,NCOLUMN1,"Horiz_pre");

        new_trial=0;
        sim_time=190;
        need_saccade=0;

        retina_r=NROW1/2;
        retina_c=NCOLUMN1/2;

        retina_rh=retina_r;
        retina_ch=retina_c;     /*previous fixation location*/

        run_early_system();
        draw_early_system();

        /*update sc_super*/
        calc_sc_superficial();
        draw_level(2,SC_SUPER,2,5,NROW2,NCOLUMN2,"SC Superficial");

        present_signal=0;
        absent_signal=0;

        time_of_last_saccade=0;
        saccade_latency=decide_next_latency();          /*when next saccade should
occur*/
        /*printf("Saccade Latency=%f\n",saccade_latency);*/
```

```
            time_of_last_gate=sim_time;
        }

        calc_input_to_salience_attn();
        gate_attn_wta_input_with_ior();

        /*increment simulation time*/
        sim_time+=2;
        /*printf("sim_time=%d\n",sim_time);*/

        if (calc_salience_attention_act() && (search_type!='f'))
        {
                calc_features_through_gate();

                if (calc_do_features_match_target())
                {
                        present_signal=1;
                        if (TO)
                                printf("Respond: Present\n");
                }
                else
                {
                        /*sim doesn't believe item was a target, so inhibit it's location*/
                        calc_ior_at(attn_r,attn_c);
                        draw_level(2,IOR_ATTN,1,4,NROW2,NCOLUMN2,"IOR (attn)");
                        clear_level(SALIENCE_ATTENTIONAL);
                        gate_attn_wta_input_with_ior();
                                /*need to calc this now so absent decision can be made with updated info*
                }
                save_2num_to_file(dwellfile,target_trial,sim_time-time_of_last_gate);
                time_of_last_gate=sim_time;
                if (TO)
                        printf("max fsw act=%f\n",calc_max_map_activity(ATTN_WTA_INPUT));
                if (calc_max_map_activity(ATTN_WTA_INPUT)<absent_threshold_attn
&& !present_signal)
                {
                        absent_signal=1;
                        if (TO)
                                printf("Respond: Absent\n");
                }
        }

        if (present_signal || absent_signal)
        {
                reaction_time=sim_time+210;
                calc_search_result_and_save_data();
                adjust_attentional_absent_threshold();
                new_trial=1;
        }

        if (calc_should_we_saccade())
        {
                /*printf("Time to saccade.\n");*/
                need_saccade=1;
```

```
    }

    if (need_saccade && make_saccades)
    {
        /*update sc deep*/
        calc_sc_deep();
        draw_level(2,SC_DEEP,3,5,NROW2,NCOLUMN2,"SC Deep");
        draw_level(2,WTA_RET,4,5,NROW2,NCOLUMN2,"WTA_Ret");

        if (TO)
        {
            printf("Performing Saccade\n");
            printf("Saccade Latency=%f\n",saccade_latency);
        }

        calc_saccadic_target();

        if (search_type=='f')
        {
            calc_search_result_and_save_data();  /*to end trial and save data*/
            new_trial=1;
            /*put a mark on the stim history map where we saccaded to*/
            draw_cross_on_stim_copy();
        }
        else
        {
            /*draw lines connecting fixations on stim_copy map*/
            draw_line_on_stim_copy();
            draw_level(1,STIM_COPY,4,0,NROW1,NCOLUMN1,"Image");
        }

        run_early_system();
        draw_early_system();

        /*update sc_super*/
        calc_sc_superficial();
        draw_level(2,SC_SUPER,2,5,NROW2,NCOLUMN2,"SC Superficial");

        time_of_last_saccade=sim_time;
        saccade_latency=decide_next_latency();      /*when next saccade should occur*/
        need_saccade=0;
    }
}


/*----------------------------Xlib Stuff----------------------------*/

void quit()
{
    XFreeGC(gdisplay, dggc);
    XCloseDisplay(gdisplay);
    exit(1);
}
```

```
get_colors()
{
    int default_depth;
    Visual *default_visual;
      XColor exact_defs[MAX_COLORS];
      int ncolors = MAX_COLORS;
      unsigned long plane_masks[1];
      unsigned long colors[MAX_COLORS];
      int i;
      XVisualInfo visual_info;
      int class;

      class = PseudoColor;
      default_depth = DefaultDepth(display, gscreen_num);
    default_visual = DefaultVisual(display, gscreen_num);
      default_cmap  = DefaultColormap(display, gscreen_num);
      if (default_depth == 1)
      {
            printf("This program needs a color monitor.\n");
            exit(0);
      }

      if (!XMatchVisualInfo(display, gscreen_num, default_depth, PseudoColor, &visual_info)) {
            if (!XMatchVisualInfo(display, gscreen_num, default_depth, DirectColor, &visual_info))
            {
                  printf("This program needs a color monitor.\n");
                  exit(0);
            }
      }

      /* got PseudoColor visual at default_depth */
      /* allocate as many cells as we can */
      ncolors = MAX_COLORS;
      while (1)
      {
            if (XAllocColorCells (display, default_cmap, False, plane_masks, 0, colors, ncolors))
                  break;
            ncolors--;
            if (ncolors = 0)
                  fprintf(stderr, "basic: couldn't allocate read/write colors\n");
            exit(0);
      }

      for (i = 0; i < ncolors; i++)
      {
            if (!XParseColor (display, default_cmap, "Black", &exact_defs[i]))
            {
                  printf("basic: color not in database");
                  exit(0);
            }
            /* set pixel value in struct to the allocated one */
            exact_defs[i].pixel = colors[i];
      }
```

```
        /* this sets the color of read/write cell */
        XStoreColors (display, default_cmap, exact_defs, ncolors);

        for (i=0;i<MAX_COLORS;i++)
                colorcell[i] = colors[i];
}


get_GC(win, gc)
Window win;
GC *gc;
{
        unsigned long valuemask = 0; /* ignore XGCvalues and use defaults */
        XGCValues values;
        unsigned int line_width = 6;
        int line_style = LineSolid;
        int cap_style = CapRound;
        int join_style = JoinRound;
        int dash_offset = 0;
        static char dash_list[] = {
                12, 24  };
        int list_length = 2;

        /* Create default Graphics Context */
        *gc = XCreateGC(display, win, valuemask, &values);

        gdisplay=display;
        ggc=gc;

        /* set line attributes */
        XSetLineAttributes(display, *gc, line_width, line_style, cap_style,
                        join_style);

        /* set dashes to be line_width in length */
        XSetDashes(display, *gc, dash_offset, dash_list, list_length);
}

void set_up_x(int argc, char *argv[])
{
    unsigned int width=7*WIN_WIDTH+8*X_BORDER,
                height=900,
                x=292, y=-22;                   /* window size and position */
        unsigned int borderwidth = 0;
        char *window_name = "Guided Search 3.0";
        char *icon_name = "GS3.0";
        char *display_name = NULL;
        unsigned char *nothing;         /*used with XChangeProperty*/
        Pixmap icon_pixmap;
        XSizeHints size_hints;
        Window win;

        /*used for setting cursor*/
        Pixmap shape, mask;
        XColor magenta_def;
```

```
Cursor curs;

/* connect to X server */
if ( (display=XOpenDisplay(display_name)) == NULL )
{
        (void) fprintf( stderr,
                        "basicwin: cannot connect to X server %s\\n",
                        XDisplayName(display_name));
        exit( -1 );
}

/* get gscreen_num size from display structure macro */
gscreen_num = DefaultScreen(display);

get_colors();

/* create opaque window */
win = XCreateSimpleWindow(display, RootWindow(display,gscreen_num), x, y,
                width, height, borderwidth, colorcell[(int)(MAX_COLORS/2.5)],
                colorcell[(int)(MAX_COLORS/2.5)]);

/* Set resize hints */
size_hints.flags = PPosition | PSize | PMinSize;
size_hints.x = x;
size_hints.y = y;
size_hints.width = width;
size_hints.height = height;
size_hints.min_width = width;
size_hints.min_height = height;

/* set Properties for window manager (always before mapping) */
XSetStandardProperties(display, win, window_name, icon_name,
   icon_pixmap, argv, argc, &size_hints);

/* Select event types wanted */
XSelectInput(display, win, KeyPressMask | PropertyChangeMask);

/* create GC for text and drawing */
get_GC(win, &gc);

XQueryColor(display, DefaultColormap(display, gscreen_num), &color);

/* Display window */
XMapWindow(display, win);

gwin=win;
dggc=gc;

XClearWindow(gdisplay,gwin);

set_all_colorscells_to_colors();

/*get keyboard focus*/
XSetInputFocus(gdisplay, PointerRoot, RevertToNone, CurrentTime);
```

```
}


void main(int argc, char *argv[])
{
        int paused=0;
        XEvent report;
        FILE *fp;

        /*decode command line flags*/
        if (argc>4)
        {
                sscanf(argv[1],"%d",&num_trials);
                sscanf(argv[2],"%s",&filename[0]);
                sscanf(argv[3],"%c",&search_type);
                sscanf(argv[4],"%d",&make_saccades);
        }
        else
        {
                printf("Usage: s #trials datafile search_type saccade?\n");
                printf("o=orientation feature\nc=color feature\nj=conjunction\nf=findlay (2
target)\n");
                printf("1=conjunction set size 1\ns=serial\nl=large set size conj\n");
                exit(0);
        }

        /*initialize random number generator*/
        set_seed_for_random_generator();

        /*routines which only need to be done once*/
        calc_cone_spacing();

        if (GRAPHICS)
                set_up_x(argc,argv);

        set_weights();

        concat2(filename,".rt",rtfile);
        concat2(filename,".dw",dwellfile);
        concat2(filename,".fi",finfile);

        /*clear the datafiles if they already exists*/
        fp=fopen(rtfile,"w");
        fclose(fp);

        fp=fopen(dwellfile,"w");
        fclose(fp);

        fp=fopen(finfile,"w");
        fclose(fp);

        /*Event Loop*/
        while (1)
        {
```

```
        if (GRAPHICS)
        if (XCheckMaskEvent(display, KeyPressMask, &report))
        {
                switch  (report.type)
                {
                case KeyPress:
                        printf("key \n");
                        paused=1-paused;
                        break;
                }
        }
        if (!paused)
                run();
        msleep(0,10000);
    }
}
```